A Cryptographic Framework for Proof of Personhood

Arka Rai Choudhuri[*] Sanjam Garg[†] Keewoo Lee[‡] Hart Montgomery[§]

Guru Vamsi Policharla[¶] Rohit Sinha[||]

Abstract

We initiate study on how to build a rigorous, cryptographic foundation for *proofs of personhood* — convincing, privacy-preserving evidence that a digital participant is a real, unique, and reputable human, optionally with authenticated attributes such as age or institutional affiliation. Towards this goal, we introduce a framework based on two types of credentials: *personhood credentials* (PHCs), issued by trusted authorities to attest to uniqueness and basic attributes, and *verifiable relationship credentials* (VRCs), issued peer-to-peer to capture reputation and real-world interactions.

We formalize ideal functionalities that capture desirable security and privacy notions for proofs of personhood, including Sybil-resistance, authenticated personhood, and unlinkability across contexts. Finally, we then give efficient cryptographic constructions that realize these functionalities by combining PHCs, VRCs, and zero-knowledge proofs. Our results suggest that a scalable, Sybil-resistant, and decentralized proof-of-personhood layer can serve as a reusable trust substrate for a wide range of online economic, social, and civic applications.

[*]zkBricks. Email: arkarai.choudhuri@gmail.com

[†]University of California, Berkeley and Exponential Science Foundation. Email: sanjamg@berkeley.edu

[‡]Ethereum Foundation. Email: keewoo.lee@ethereum.org

[§]Linux Foundation. Email: hmontgomery@linuxfoundation.org

[¶]University of California, Berkeley. Email: guruvamsip@berkeley.edu

[||]Swirlds Labs. Email: sinharo@gmail.com

Contents

1 Introduction 4
- 1.1 Our Contribution: Cryptography for Proof of Personhood . . . 5

2 Technical Overview 7
- 2.1 Modeling Mis-issuance of Credentials . . . 8
- 2.2 Defining Security for Proofs of Personhood . . . 8
- 2.3 Construction of Proofs of Personhood . . . 11
- 2.4 Black-box Construction . . . 13

3 Preliminaries 15
- 3.1 Bilinear maps . . . 15
- 3.2 Commitment Schemes . . . 16
- 3.3 Signature Schemes . . . 16
- 3.4 Pseudorandom Functions . . . 17
- 3.5 Non-interactive Zero-knowledge . . . 17
- 3.6 Groth-Sahai proofs . . . 18
- 3.7 Hiding KZG Polynomial Commitments . . . 19

4 Personhood Oracle 20
- 4.1 Discussion . . . 22

5 Definition 23
- 5.1 Security . . . 24
- 5.2 Interpreting the Ideal Functionality . . . 26
- 5.3 Impossibility of Full Unlinkability . . . 27
- 5.4 Example f - Supporting Revocation . . . 28

6 Vouchable Credentials 29

7 Proof of personhood 31
- 7.1 Construction I: realizing $\mathcal{F}_{\mathsf{PoP}}$ . . . 31
  - 7.1.1 Components . . . 31
  - 7.1.2 Proof of Knowledge Languages . . . 32
  - 7.1.3 Construction . . . 32
  - 7.1.4 Security Proof . . . 33
- 7.2 Construction II: realizing $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ . . . 39
  - 7.2.1 Components . . . 40
  - 7.2.2 Proof of Knowledge Languages . . . 40
  - 7.2.3 Construction . . . 41
  - 7.2.4 Security Proof . . . 42

8 Blackbox Vouchable Credentials 46
- 8.1 Vouchable Credential Instantiation . . . 47
- 8.2 Security Proof . . . 49

9 Blackbox Proofs of Personhood 52
- 9.1 Instantiating Proofs of Personhood . . . 53

10 Evaluation 54

A Related Work 62

B Applications **62**
B.1 AI Agent Reputation 62
B.2 Open Source Maintainer Provenance 63
B.3 Decentralized Business Networks 64

C Generic Construction of Vouchable Credentials **64**
C.1 Construction 65

1 Introduction

"On the Internet, nobody knows you're a dog," the famous New Yorker comic from 1993 goes [Ste93]. But this lighthearted early meme belied a significant problem that the digital world brought: how do you know that you're interacting with a real human online? Or, even better, a human of the appropriate age or with the correct credentials?

The first major advancement in so-called *proofs of personhood*, even if it was not called as such, was the seminal CAPTCHA paper [vABHL03] (Eurocrypt '03). CAPTCHAs relied on the Turing Test [Tur50] being hard for at least some problems; the main idea was to provide a (potential) person with a short task that should be very easy for a human but hard for a machine. For a while, CAPTCHAs were very effective at stopping spam and bots from performing various nefarious activities on the web.

However, with the advancements in machine learning today, CAPTCHAs are essentially completely broken. There is a large line of recent work on breaking all kinds of CAPTCHA systems using LLMs and other ML techniques [DOL+25, TLL+25, PVW24]. There have even been documented attacks on websites using AI tools to break CAPTCHAs, such as Akirabot [Lak25], which is an OpenAI-based tool used to target 420,000 sites with spam by bypassing CAPTCHAs. We believe it should be clear that we can no longer rely on humans being "smarter" than bots or agents for proofs of personhood.

So how can we prove personhood in a world where bots are as smart as humans? Most solutions today are very ad-hoc and actively detrimental to user privacy. While we could fill a paper with such solutions, we will only focus on a few here. As an example, mobile driver's licenses (mDLs) are becoming more popular in the United States. However, to prove (active) personhood, the standard allows the driver's license verifier to query the Department of Motor Vehicles (DMV) server for a revocation check! This means the DMV can potentially see and record every verification done, which is a huge privacy issue [Sin25, Bur25]. There is a large list of organizations and experts that have spoken out against this practice [Und], but so far there has been little concrete action by mDL issuers.

These privacy risks are not imaginary. Discord recently was compromised and had over 70,000 government IDs of users stolen [Goo25]. Very recently, Discord sparked backlash by announcing that all users would have to verify their age using a government ID to see any kind of "adult" content [Bel26]–their way of proof of personhood. Why should users trust Discord with their personal information at this point?

The one institution that has most directly tried to build a proof of personhood has been World, with Worldcoin [Fou26]. However, their global biometric-based system has received a considerable amount of backlash. For instance, EPIC [Cen23] stated, "Worldcoin is a potential privacy nightmare... Mass collections of biometrics like Worldcoin threaten people's privacy on a grand scale, both if the company misuses the information it collects, and if that data is stolen... We urge regulatory agencies around the world to closely scrutinize Worldcoin." Regulators everywhere have taken note: governments in Europe like Spain and Bavaria have required World to delete user biometric data [Reu24b], with the Bavarian government [ftPS25] stating, "as a result of our investigation of the processing of the Worldcoin Foundation, we find that the Worldcoin Foundation has violated the General Data Protection Regulation (GDPR) as described in detail below." China has issued warnings against Worldcoin [oSS25], and Hong Kong regulators [Reu24a] ordered WorldCoin to cease all operations in the country, dubbing its data collection as "unnecessary and excessive." Finally, even places like Kenya, which in theory should benefit most from Worldcoin, have had it shut down, with a reporter [Bil26] noting "From the Constitution to statutory law, every imaginable violation occurred," about the operation of Worldcoin in Kenya. This motivates the following important question:

How does one prove personhood online?

We note that none of these systems rigorously formalize what a proof of personhood actually is, or give cryptographic guarantees around it. We will obviously do this, but before we do, we want to point out that this sort of notion appears in a wide variety of other applications, often masquerading as a different kind of problem.

One of the most notorious and potentially dangerous cyberattacks in recent years was the so-called XZUtils attack [Jam25, Kas24]. A maintainer, calling themselves Jia Tan, targeted an undermaintained project critical to the Linux kernel, and started contributing. After this individual (or group of individuals) became an accepted maintainer, they uploaded malware to the codebase. This malware was barely caught at the last second, mitigating what could have been an extremely severe compromise in the Linux kernel. This sort of attack is becoming widespread even outside of open source software. In particular, North Korean tech operatives frequently attempt to use ML tools to gain remote work jobs for tech companies in the US and Europe [Kap25]. According to Mandiant [MN25], nearly every Fortune 500 CIO interviewed admitted to hiring a North Korean engineer.

There are other issues with open-source code projects too: unscrupulous developers are contributing so-called "AI slop" to projects in a highly automated fashion, burdening project maintainers with reviewing large amounts of poorly written code. This has become enough of a problem that, very recently, Mitchell Hashimoto, the CEO of Hashicorp, personally built a reputation system for maintainers called Vouch [Has] to aim at solving this problem, which has gotten quite a bit of attention. Unfortunately, this system lacks many of the guarantees we need, including privacy, but is a start in the right direction. So:

How does one prove (reputable) personhood online?

The Internet enabled unprecedented communication, collaboration, and commerce. However, this progress always implicitly relied on our ability to distinguish humans from machines and, equally importantly, to verify the identity of the human on the other side of a digital interaction. Today's mechanisms for establishing this trust online, range from passwords to centralized identity providers, and both have proven insufficient. Password-based systems, despite their simplicity, create systemic vulnerabilities: services storing passwords become high-value targets for attackers; credential reuse across platforms is widespread; and large-scale data breaches remain persistent and inevitable. Centralized identity providers introduce additional structural risks, concentrating power and creating single points of failure. Moreover, centralized systems prevent trust from being portable across contexts. A driver's reputation on a ridesharing platform cannot be transferred to a vacation rental service, even when the underlying trust relationship is genuine. These lock-in effects represent a substantial loss of value for users and fragment trust across digital ecosystems. This motivated the following question:

How does one prove (authenticated) personhood online?

In summary, our thesis is a scalable, Sybil-resistant, decentralized proof of reputable and authenticated personhood can serve as the basis of a trust layer for the Internet.

1.1 Our Contribution: Cryptography for Proof of Personhood

In this work, we combine *personhood credentials (PHCs)* and *verifiable relationship credentials (VRCs)* and efficient zero-knowledge proofs on top of them to form a coherent system for a proof of personhood.

PHCs are issued by trusted authorities and provide evidence that an individual is a real, unique human. These authorities can include educational institutions issuing student IDs, governments issuing passports or driver's licenses, employers certifying employment, or ride-sharing platforms issuing driver credentials. Any organization can serve as a *credential-issuing authority*, provided it ensures that each person receives at most one PHC and follows revocation protocols when necessary. PHCs may also encode additional structured information, such as age or role, to support specific applications.

VRCs are issued peer-to-peer between individuals and capture real-world interactions or trust relationships. For example, a passenger and driver on a ride-sharing platform may issue VRCs to one another after a completed trip. VRCs can also carry endorsements or qualitative feedback, such as a colleague vouching for reliability or a landlord attesting to tenant trustworthiness.

By combining multiple PHCs from different authorities and VRCs from peers, we create a robust representation of identity, personhood, and other relevant attributes. Crucially, individuals need not reveal all their credentials to every verifier. Using the zero-knowledge proofs [GMR89], individuals can prove properties such as being over a certain age or having sufficient endorsements without exposing underlying personal data. This approach allows verification to be privacy-preserving, cryptographically sound, and usable at scale.

An ecosystem of users, companies, and even AI agents with these credentials would provide the backbone of a decentralized trust layer that could enable the online world to continue to grow and flourish in the age of AI.

Our Results. We make the following contributions. First, we formalize a proof-of-personhood system in the real/ideal world paradigm by defining an ideal functionality $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$. The functionality captures the issuance of personhood credentials (PHCs), the creation of verifiable relationship credentials (VRCs), and the ability to prove predicates over collections of such credentials. It provides: (i) *soundness*, ensuring that any valid show proof must be supported by underlying credentials and attestations; (ii) *privacy*, ensuring that only information explicitly specified by the predicate is revealed; and (iii) *unlinkability*, ensuring that credential issuance and attestation interactions cannot be linked across issuers or contexts beyond what is explicitly revealed.

We further extend the model to capture imperfect PHC issuance by incorporating global validation oracles. These oracles model external validation procedures that may err with bounded probability, allowing us to reason about settings in which issuers can be deceived by adversarial users. In this model, when a show proof aggregates credentials and attestations from multiple independent issuers, the confidence in the resulting claim increases as additional honest sources are incorporated.

Finally, we give a protocol realizing the functionality. To do so, we introduce a new primitive, *vouchable credentials*, which augment standard credentials with the ability to produce succinct proofs tied to revealed attributes. We obtain the following.

Theorem 1 (informal). *Assuming the security of pseudorandom functions, non-interactive zero-knowledge proofs, and vouchable credentials, there exists a protocol that securely realizes* $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ *in the presence of global validation oracles.*

We also consider a weaker functionality $\mathcal{F}_{\mathsf{PoP}}$ in which unlinkability guarantees are removed, capturing deployments where persistent identities are acceptable and efficiency is prioritized.

Theorem 2 (informal). *Assuming the security of non-interactive zero-knowledge proofs and vouchable credentials, there exists a protocol that securely realizes* $\mathcal{F}_{\mathsf{PoP}}$ *in the presence of global validation oracles.*

While vouchable credentials can be constructed generically from any credential scheme together with non-interactive zero-knowledge proofs, we additionally present a concrete instantiation designed for efficiency that makes only black-box use of underlying cryptographic primitives.

Theorem 3 (informal). *There exists a vouchable credential scheme in the generic group model that makes only black-box use of cryptographic primitives.*

We also provide experimental evaluation results, implementing the construction using both our black-box vouchable credential scheme and general-purpose zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs).

Roadmap. In the remainder of the paper, we present a technical overview in Section 2, then define the personhood oracle and formal security model in Sections 4 and 5. We next introduce our abstraction of vouchable credentials in Section 6. We provide both a generic instantiation of vouchable credentials in Appendix C, and a black-box construction in Section 8. Using vouchable credentials as an underlying tool, we present our proof-of-personhood constructions (base and unlinkable variants) in Sections 7.1 and 7.2 and conclude with experimental results in Section 10.

2 Technical Overview

In this section, we present a high-level overview of the ideas underlying our work. Our goal is to construct a proof-of-personhood system in which users can accumulate attestations from credential holders and later prove aggregate properties of these attestations, while formally accounting for mis-issuance and providing strong privacy and unlinkability guarantees. To motivate the technical ideas, we begin by describing a typical flow for issuing and using credentials.

A user, Alice, wishes to obtain a certification of her personhood from some authority. In practice, this means that Alice presents a collection of attributes to an issuer who verifies them and produces a signed credential. For example, Alice may present the following attributes:

```
att =
{
   Name = Alice
   Age = 35
   Occupation = Cryptographer
   Phone = 555-3141
}
```

We call an authority capable of issuing such credentials a *personhood credential (PHC) issuer*. In our framework, a PHC is simply a credential issued by an authority that attests to attributes associated with a real-world identity; personhood arises from aggregating such credentials and attestations rather than from any single issuer. To obtain a PHC credential, Alice presents herself along with these attributes to the issuer. She also includes a public key, for which she proves knowledge of the corresponding secret key, so that the credential can later be used online.

While ownership of the public key can be verified cryptographically, the veracity of the remaining attributes is established through an inherently ad-hoc process. The issuer verifies Alice's identity and attributes according to its own procedures, for instance, by inspecting identification documents or meeting Alice in person. These procedures are not standardized and vary widely across issuers. We will return to this point shortly, as it is an important consideration for our model.

In practice, users implicitly rely on social and institutional notions of trust when deciding which issuers to accept. For example, a government agency such as a DMV is typically considered a more reliable PHC issuer than a restaurant issuing a credential based on frequent visits. This distinction reflects the perceived reliability of each issuer's verification process.

Our goal is to democratize the process of issuing personhood credentials while still formally accounting for the risk that an issuer may incorrectly certify a user. Rather than assuming a fixed set of highly trusted authorities, we aim to allow a broader set of issuers while explicitly parameterizing the risk of mis-issuance. This leads to a model in which trust can be accumulated through attestations and relationships between credential holders, forming the basis for our proof-of-personhood construction.

A credential holder can then use their PHC credential to attest to properties of other users. For example, Alice can vouch for another user, Bob, by issuing an attestation stating `‘‘Bob is a cryptographer’’`, revealing the relevant attribute "`Occupation = Cryptographer`" and the PHC issuer (say, her university) in the process. In this instance, Alice is a *verifiable credential (VRC) issuer*. The issued VRC can be verified against the PHC issuer's public key.

Finally, Bob can collect VRCs from many credential holders (say N), each of whom attests to the statement `‘‘Bob is a cryptographer’’`. Using these VRCs, Bob generates a *show proof* asserting that `‘‘there exist` N `credential holders who attest that Bob is a cryptographer’’`, without revealing the identities of the individual VRC issuers.

While the above description captures the functional workflow of our system, there are several security considerations that arise when combining multiple PHCs from different authorities and VRCs from peers. In particular, we must ensure that aggregating credentials and attestations yields a robust notion of identity and personhood, even when some issuers and users behave incorrectly.

In the remainder of this overview, we first describe how we model the mis-issuance of credentials. We then present our ideal functionality and the security guarantees it provides, before outlining the high-level ideas behind our construction.

2.1 Modeling Mis-issuance of Credentials

A primary point of failure in the system occurs at the very start, when a user obtains a credential from a PHC issuer. Each issuer may employ arbitrary and potentially imperfect procedures when deciding whether to issue a credential. Since issued credentials are often treated as statements of fact about a user, it is important to explicitly parameterize our confidence in each issuer's verification process.

Traditional models of credential issuance allow the issuer to abort during issuance, implicitly capturing the possibility that a credential is refused. However, this abstraction does not accurately reflect the real world, where an adversarial user may attempt to fool an issuer into issuing a credential based on incorrect attributes.

We therefore adopt a different approach that separates the validation of attributes from the act of issuing the credential. This allows us to model the risk of mis-issuance explicitly, rather than treating issuance as an all-or-nothing event controlled solely by the issuer.

To formalize this, we associate with each issuer I a *personhood attribute validation (PAV) oracle* O_I. When a user U requests a credential for attributes att, the user sends att to the issuer I. The issuer forwards att to O_I, while the user's identity is implicitly provided via the interaction itself. The oracle then returns a bit indicating whether it considers att valid for that user. The issuer proceeds with credential issuance only if the oracle returns 1. The PAV oracle models the issuer's real-world validation process and is treated as an external primitive rather than something simulated by the protocol. In particular, it is available in both the real and ideal worlds.

For each issuer I we associate a confidence parameter ε_I, which captures the probability that the oracle incorrectly validates attributes. Formally, we require that

$$\Pr\left[\ \mathsf{O}_I(U, \mathsf{att}) = 1 \mid \mathsf{att} \text{ is not valid for } U\ \right] \leq \varepsilon_I.$$

This parameter ε_I allows us to reason about systems that aggregate credentials from multiple issuers with varying reliability. Looking ahead, an important design choice in our model is that the verification of VRCs and *show proofs* reveals the identities of the issuers whose credentials underlie them. This allows a verifier to determine the level of confidence it assigns to a proof based on the issuers involved and their associated error parameters.

With this modeling choice, we can capture a wide range of real-world systems, including deployments such as Worldcoin [Fou26], by associating an appropriate confidence parameter with each issuer. Importantly, regardless of the underlying issuer, our system provides strong guarantees of robustness and privacy. These guarantees hold independently of the issuer's reliability, except for privacy leakage that may arise directly from the issuer itself (e.g., if the issuer learns or reveals user attributes during credential issuance).

In the technical section we further refine this model to capture the fact that a (financially) powerful adversary may have a higher probability of fooling an issuer. In such settings, the parameter ε_I is better viewed as a *confidence function* rather than a fixed constant. We refer the reader to the technical section for details.

2.2 Defining Security for Proofs of Personhood

We model a proof-of-personhood system via the following components. We give an informal description here; the formal definitions appear in Section 5.1.

$\mathsf{IssuePHC}$: An interactive protocol between a user U and an issuer I. At the end of the protocol, the user obtains a personhood credential cred on a set of attributes att. During the protocol, the user and issuer provide their respective inputs to the validation oracle O_I, which determines whether the attributes are valid for the user. The issuer proceeds with issuance only if the oracle returns 1.

$\mathsf{IssueVRC}$: An algorithm run by a credential holder U possessing a credential cred on attributes att. The algorithm produces a verifiable relationship credential (VRC) vrc for a recipient V, attesting to a statement st within a context ctx. During this process a subset of attributes $\mathsf{att}' \subseteq \mathsf{att}$ is revealed, together with the identity of the underlying PHC issuer I whose public key will be used to verify the VRC.

$\mathsf{VerifyVRC}$: A deterministic algorithm that checks the validity of a VRC vrc for statement st in context ctx with respect to issuer I's public key and the revealed attributes att'.

$\mathsf{ShowVRC}$: An algorithm run by a user U that takes a collection of VRCs issued to U together with a predicate f. It produces a *show proof* asserting that the provided VRCs satisfy the predicate f.

$\mathsf{VerifyShow}$: A deterministic algorithm that verifies a show proof with respect to the issuer public keys corresponding to the VRCs used in its construction.

We will elaborate on the necessity of context strings during VRC issuance shortly. Intuitively, a context represents a domain of interaction, such as `work` or `restaurant`. Contexts allow attestations issued within the same context to be linkable when necessary, while preventing linking across different contexts.

Ideal Functionality. To provide strong security guarantees, we define an ideal functionality $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$, requiring security in the real/ideal world paradigm. While the formal description is provided in Section 5.1, we briefly describe the interaction of the participants of the system with $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$.

The functionality maintains two lists, S_C and S_A, tracking all of the issued credentials and relationship attestations. It further maintains a mapping of users and ephemeral keys $\mathsf{keyMaps}$.

Credential Issuance: A user U requests a credential for attributes att from I. The functionality samples a random ephemeral string ξ and passes the request to I on behalf of ξ (hiding U). If I approves, the functionality stores $(U, \xi, \mathsf{att}, I)$ in S_C and sends ξ to U and indicates that the credential was issued.

Context Key Issuance: A user U requests a context-specific key for context ctx and ephemeral key ξ. The functionality samples a random ephemeral string η. It then adds $(U, \xi, \mathsf{ctx}, \eta)$ to $\mathsf{keyMaps}$, returning η to U.

Relationship Attestation Issuance: A user U requests a relationship attestation on a statement st in context ctx from a user identified by context key η. The functionality identifies the corresponding user V from $\mathsf{keyMaps}$ and forwards the request to V using a fresh ephemeral handle ρ, hiding the requester's identity.

If V approves the request and specifies revealed attributes att' together with an issuer I, the functionality checks that V holds a credential issued by I whose attributes include att'. If so, it samples a fresh attestation identifier id, records $(\mathsf{id}, U, \rho, V, \mathsf{ctx}, \mathsf{st}, \mathsf{att}', I)$ in S_A, and returns to the requester the revealed attributes att', issuer identity I, the ephemeral handle ρ, and the identifier id.

Attestation Showing: A user U may request to prove a predicate f over a collection of attestations identified by $\mathsf{id}_1, \ldots, \mathsf{id}_N$. The functionality retrieves the corresponding attestations from S_A and evaluates f. If the predicate holds, it sends to the verifier the predicate f, the associated statements, and the identities of the issuers involved. The identities of the individual attesting users remain hidden.

The ephemeral values above serve as unlinkable handles that allow the functionality to track relationships without revealing long-term identities.

In the technical section, we additionally formalize the ideal-world execution in the presence of the PAV oracles. Unlike models in which such oracles appear only in the real world, the PAV oracles in our setting are *global* and are accessible to parties in both the real and ideal worlds. This ensures that the ideal functionality does not subsume the behavior of the validation mechanism, and that any guarantees derived from the PAV oracles are preserved across both executions.

Interpreting the ideal world. The ideal functionality provides a clear interpretation of what a successful show means. Each credential used in a show must ultimately trace back to an issuance approved by the corresponding issuer's validation oracle. When issuers are honest, false attributes are accepted only with the oracle's one-sided error probability. Because the validation oracles of different issuers are sampled independently, each additional honest issuer involved in a show increases the verifier's confidence that the underlying attributes are truthful. Credentials originating from corrupted issuers provide no such guarantee, but they also do not affect the guarantees contributed by honest issuers.

Security Guarantees. In addition to correctness, the following security guarantees are provided from any system that securely realizes $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$

Soundness: No adversarial user can convince an honest user or verifier of a statement that is not supported by valid underlying credentials and attestations. In particular:

- An adversary cannot produce a valid VRC without possessing a corresponding underlying PHC credential.
- An adversary cannot produce a valid show proof for a predicate f unless the underlying attestations satisfy f in the ideal functionality.

Privacy: The system guarantees the following forms of privacy:

- *VRC issuance attribute hiding:* All attributes of the underlying PHC that are not explicitly revealed during VRC issuance remain hidden from the recipient and any outside observer.
- *Show-proof VRC hiding:* A show proof reveals only the information specified by the predicate f and the associated issuers. The individual VRCs and the identities of the users who issued them remain hidden.

Unlinkability: The system provides the following unlinkability guarantees:

- *PHC receiver unlinkability:* Even colluding malicious issuers cannot link multiple PHC credentials obtained by the same user.
- *VRC receiver unlinkability:* Even colluding malicious users cannot link multiple VRCs received from the same underlying credential holder within a context unless such linking is explicitly allowed.
- *VRC issuer cross-context unlinkability:* Colluding users cannot link VRCs issued by the same credential holder across different contexts.

Furthermore, even colluding adversarial issuers and users cannot link a user's PHC credential to the VRCs they later issue, except where explicitly revealed by the protocol (e.g., through issuer identities).

Base functionality. In addition to $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$, we define a *base* functionality $\mathcal{F}_{\mathsf{PoP}}$ that does not provide unlinkability guarantees. This functionality captures deployments in which users are willing to operate under persistent identities and where unlinkability across interactions is not a design goal.

In $\mathcal{F}_{\mathsf{PoP}}$, all users are associated with long-term identities that are revealed during both PHC and VRC issuance. The functionality does not generate ephemeral handles or context-specific pseudonyms. Instead, whenever a credential is issued or a relationship attestation is created, the real identity of the user is included in the message delivered to the counterparty.

This variant models settings in which efficiency or simplicity is prioritized over unlinkability. In particular, removing unlinkability requirements allows for more efficient constructions, and in some deployments, unlinkability is not necessary.

Impossibility of full unlinkability. We now explain the need for context-dependent keys and, consequently, the limitation to context-dependent unlinkability. One might hope to strengthen $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ to achieve *full* unlinkability, in which no contexts exist and a VRC issuer may simply request arbitrary pseudonyms that can be used to issue VRCs. In such a functionality, the system would maintain a mapping from users to all pseudonyms issued to them, allowing it to reason about whether two VRCs originate from the same underlying user.

However, no real-world protocol can realize this stronger notion. To see this, consider a predicate f that checks whether two VRCs used in a show proof originate from distinct underlying users. In the ideal world, the functionality can enforce this requirement because it internally maintains the mapping between users and pseudonyms. When a show proof is requested, it can simply check whether the corresponding underlying users are distinct.

In the real world, however, any protocol achieving full unlinkability must ensure that pseudonyms issued to the same user are indistinguishable from pseudonyms issued to different users. Consequently, a verifier cannot distinguish between the following two scenarios: (i) two distinct users each issuing a VRC under separate pseudonyms, and (ii) a single user generating two pseudonyms and issuing both VRCs. Since these scenarios are computationally indistinguishable, no real protocol can enable a verifier to reliably enforce the predicate f while preserving full unlinkability.

This rules out achieving full unlinkability in our setting. We therefore adopt the more practical notion of *context-dependent unlinkability*, in which unlinkability holds across different contexts but allows linking within the same context when required for functionality. A formal description of this impossibility is presented in Section 5.3.

2.3 Construction of Proofs of Personhood

We now describe our construction of proofs of personhood. We proceed in two steps. First, we construct a system that securely realizes the base functionality $\mathcal{F}_{\mathsf{PoP}}$. We then show how to upgrade this construction to additionally realize $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ by incorporating context-dependent unlinkability.

At a high level, our construction combines a credential scheme with zero-knowledge proofs. However, because we aim to securely realize $\mathcal{F}_{\mathsf{PoP}}$, we require a stronger notion of security than is typically needed for credential systems. In particular, the underlying credential scheme must satisfy additional properties that allow credentials to be safely used as building blocks inside the larger protocol without compromising soundness or privacy.

A credential scheme consists of an interactive protocol Issue between a user U and an issuer I, where the issuer is identified by a public key ipk. At the end of this interaction, the user obtains a credential cred on a set of attributes att agreed upon by both parties. For intuition, the reader may think of the credential as a signature on a commitment to the attributes. In a typical deployment, the user can later use cred to reveal a subset of attributes $\mathsf{att}' \subseteq \mathsf{att}$ and prove that these attributes are certified under the issuer's public key.

In our setting, credentials must support an additional operation: using a credential to generate attestations about other users. To capture this capability, we introduce the notion of *vouchable credentials*. A vouchable credential scheme extends a standard credential system by allowing a credential holder to produce a *vouch* for a statement using their credential.

We formalize the required properties under this abstraction of *vouchable credentials*. By clearly isolating the guarantees we need from the underlying credential scheme, the security proof of our overall construction becomes modular and conceptually simpler. Moreover, this abstraction lets us focus on designing and instantiating more efficient vouchable credential schemes independently of the surrounding proof-of-personhood protocol.

Formally, in addition to the issuance protocol Issue, a vouchable credential scheme provides two additional algorithms: Vouch and Verify. The Vouch algorithm allows a credential holder to generate a vouch for a statement while revealing only a selected subset of attributes, and Verify allows anyone to check the validity of the resulting vouch under the issuer's public key.

Each user U is assumed to possess a long-term key pair $(\mathsf{pk}_U, \mathsf{sk}_U)$. To bind this key to the issued credential, the public key pk_U is included as one of the credential attributes. During VRC issuance this attribute is revealed, allowing recipients to associate all vouches issued by the same credential holder with a consistent public key and thereby build trust relative to that key.

We now describe the components. Note that in our base construction, we do not need to use contexts for the issuance and verification of the VRC.

$\underline{\mathsf{IssuePHC}(\langle I(\mathsf{isk}), U(\mathsf{ipk}, \mathsf{att}, \mathsf{pk}_U, \mathsf{sk}_U)}$

1. User U demonstrates to I knowledge of the secret key corresponding to pk_U, and sends att to the I.
2. U and I interact with O_I to determine if I should proceed with the issuance of a vouchable credential.
3. U and I run Issue of the vouchable credential with the attributes $\mathsf{pk}_U||\mathsf{att}$, where the I input is its secret key isk.
4. U obtains cred.

$\underline{\mathsf{IssueVRC}(\mathsf{cred}, \mathsf{pk}_U, \mathsf{att}', \mathsf{st}, \mathsf{pk}_V)}$

1. Bind the statement to the recipient V as $\mathsf{st}' = \mathsf{pk}_V||\mathsf{st}$.
2. Compute a vouch $\mathsf{vouch} \leftarrow \mathsf{Vouch}(\mathsf{cred}, \mathsf{pk}_U||\mathsf{att}', \mathsf{st}')$.
3. Output the VRC as $(\mathsf{vouch}, \mathsf{att}', \mathsf{ipk}, \mathsf{st}, \mathsf{pk}_U)$ to V.

$\underline{\mathsf{VerifyVRC}(\mathsf{pk}_U, \mathsf{vouch}, \mathsf{att}, \mathsf{ipk}, \mathsf{st}, \mathsf{pk})}$

1. User U outputs the result of the the vouch verification algorithm $\mathsf{Verify}(\mathsf{ipk}, \mathsf{pk}||\mathsf{att}, \mathsf{st}||\mathsf{pk}_U, \mathsf{vouch})$.

$\underline{\mathsf{ShowVRC}(\mathsf{pk}_U, f, \{\mathsf{vouch}_i, \mathsf{att}_i, \mathsf{ipk}_i, \mathsf{st}_i, \mathsf{pk}_i\}_i)}$

1. Set the NIZK statement $\mathsf{x} = (\mathsf{pk}_U, f, \{\mathsf{st}_i, \mathsf{ipk}_i\}_i)$.
2. Set the NIZK witness $\mathsf{w} = (\{\mathsf{vouch}_i, \mathsf{att}_i, \mathsf{pk}_i\}_i)$.
3. Compute the $\mathsf{nizk} \leftarrow \mathsf{NIZK.Prove}(\mathsf{x}, \mathsf{w})$.
4. Output x and nizk.

$\underline{\mathsf{VerifyShow}(\mathsf{pk}_U, f, \mathsf{nizk}, \{\mathsf{st}_i, \mathsf{ipk}_i\}_i)}$

1. Set the NIZK statement $\mathsf{x} = (\mathsf{pk}_U, f, \{\mathsf{st}_i, \mathsf{ipk}_i\}_i)$.
2. Output the result of the NIZK verification $\mathsf{NIZK.Verify}(\mathsf{x}, \mathsf{nizk})$.

The NIZK checks: (i) each of the input vouches vouch_i are valid with respect to the vouch verification algorithm; and (ii) the predicate $f(\{\mathsf{st}_i, \mathsf{att}_i, \mathsf{ipk}_i\}_i)$ outputs 1.

Proving Security and Vouchable Credential Properties. To prove security we work in the real/ideal paradigm and construct a simulator that reproduces the view of any real-world adversary while interacting only with the ideal functionality. Because the ideal functionality enforces strong correctness and privacy guarantees, the simulator must be able to faithfully emulate all protocol messages using only the information revealed by the functionality.

This requirement places strong demands on the underlying vouchable credential scheme. In particular, when the simulator receives from the ideal functionality only a subset of attributes corresponding to an attestation issued by an honest party, it must be able to *simulate* a valid vouch using only this revealed information. Conversely, when corrupted parties generate credentials or vouches using malformed keys or inconsistent attributes, the scheme must still ensure soundness, which necessitates strong unforgeability and extraction guarantees.

We formalize these requirements through the abstraction of *vouchable credentials*. We defer the precise definitions to the technical sections, but emphasize that identifying these properties explicitly allows our proof to be modular and may be of independent interest beyond the present construction.

Adding unlinkability. To upgrade the base system to additionally achieve unlinkability, we replace each user's stable public identifier with a family of *pseudonyms* derived from a secret PRF key K_U. Concretely, whenever the protocol requires a user identifier or public key, U derives a fresh pseudonymous identifier as a PRF output, using the issuer identity and/or the context as input.

- *PHC issuance.* When interacting with a PHC issuer I, U derives an issuer-specific pseudonym $k_{U,I} := \mathsf{PRF}_{K_U}(I)$.
- *VRC issuance in a context.* When issuing a VRC in context ctx using a PHC under issuer I, U derives a context-specific pseudonym $k_{U,I,\mathsf{ctx}} := \mathsf{PRF}_{K_U}(I \,\|\, \mathsf{ctx})$.
- *VRC receipt / interaction-specific pseudonym.* When interacting with a counterparty identified by a (context) pseudonym k', U derives an interaction-specific pseudonym $k_{U,k'} := \mathsf{PRF}_{K_U}(k')$.

These derived pseudonyms intentionally differ across issuers, contexts, and interactions, preventing linkability from the identifiers alone. At the same time, the protocol must still convince verifiers that the various pseudonyms used in credential issuance, VRC issuance, and showing are all consistent with a single underlying credential holder when required by the functionality.

We achieve this by shifting the relevant consistency checks into zero-knowledge proofs. Rather than revealing linkable structure, the VRC issuance proof shows that the attester possesses a valid PHC credential and that the pseudonym used to issue the VRC is correctly derived (via the PRF) from the same underlying secret as the pseudonym bound to the PHC. Similarly, the show proof checks in zero knowledge that the pseudonyms embedded in the underlying VRCs are consistent with the pseudonym revealed for the show, while hiding any additional information. The zero-knowledge property ensures that these checks do not introduce new linkability beyond what is intentionally revealed.

We next present an instantiation of vouchable credentials, that only make black-box use of cryptographic primitive, i.e. avoids using a general purpose NIZK to obtain a vouch from a credential scheme.

2.4 Black-box Construction

Next we describe our blackbox-construction of vouchable credentials. In essence we will be instantiating the generic construction of proofs of personhood Fig. 2 with carefully designed vouchable credentials and NIZK proof system. Before we describe our construction, we first outline some guiding principles that we follow in order to obtain an efficient construction. Let's begin by working backwards with goal of a black-box construction:

- We want a black-box prover for the relation in $\mathsf{ShowVRC}$ – this essentially proves knowledge of valid vouches under some issuer public key and satisfaction of the predicate f. We already appear to be in trouble as vouches are effectively signatures, and the predicate f is an arbitrary circuit over a field (say) – naively using a 'monolithic' proof system (such as a zkSNARK) would require us to be non-black-box in the signature scheme.

  But observe that f only acts on the attributes of the vouches, not the vouches themselves. As a result we can use two proof systems – one for proving knowledge of vouches and another for proving the attributes satisfy the predicate f. Together with a 'linking-proof' we can show consistency of attributes between the two proof systems. A similar strategy was used in [CGM16, AGM18, CFQ19] to prove 'mixed' statements involving different algebraic structures.

- Next, we need to design a vouchable credential that comes with an efficient proof of knowledge. Given that vouchable credentials imply a signature scheme due to the vouch non-malleability guarantee Section 6, the proof system proves something at least as powerful as a signature. It will serve us well to choose our proof system accordingly.

Discrete logarithm based signatures involve random oracles and therefore can be ignored. Thus, we turn to pairings where the situation is much better. A long line of work on Structure Preserving Signatures (SPS) starting from [AFG+10] has shown that it is possible to design signatures whose verification is *algebraic* – only involving pairing product equations which can then be combined with the Groth-Sahai proof system [GS08] to prove knowledge of a signature scheme. Of course, Vouchable Credentials are much more powerful than signatures, and we have not yet made any progress towards building them so it's not clear if Groth-Sahai will be sufficient. But we can now hope to design analogous *Structure Preserving* Vouchable Credentials that admit algebraic verification.

- The final piece of the puzzle is the personhood credential from the issuer. Again, it's at least as strong as a signature because of the unforgeability guarantee which needs to be proved in zero-knowledge during $\mathsf{ShowVRC}$ and potentially in $\mathsf{IssueVRC}$. Using the same justification as above, we gravitate towards designing structure preserving personhood credentials.

Based on the guidelines above, we now have a template for vouchable credentials:

$\underline{\mathsf{Init}(1^\lambda, 1^\ell)}$: I samples a signing key of a structure preserving signature scheme $(\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{SPS.Gen}(1^\lambda)$.

$\underline{\mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle}$:

- U samples a key pair $(\mathsf{sk}_U, \mathsf{pk}_U) \leftarrow \mathsf{SPS.Gen}(1^\lambda)$
- U creates a hiding commitment to their attributes att: $\mathsf{com}_{\mathsf{att}}$.
- I signs $\mathsf{pk}_U || \mathsf{com}_{\mathsf{att}}$ which acts as the credential cred.

$\underline{\mathsf{Vouch}(\mathsf{st}, \mathsf{att}, \mathsf{idx}, \mathsf{cred})}$:

- To vouch for st, with attributes $\mathsf{att}' \subset \mathsf{att}$, the user signs st using sk_U to produce σ_{st}.
- Additionally, they prove $\mathsf{com}_{\mathsf{att}}$ opens to the claimed attributes $\pi_{\mathsf{att}} = \mathsf{Open}(\mathsf{com}_{\mathsf{att}}, \mathsf{idx}) \rightarrow \mathsf{att}'$.
- The vouch is then set to be $\pi = (\pi_{\mathsf{att}}, \mathsf{cred}, \sigma_{\mathsf{st}})$.

$\underline{\mathsf{Verify}_{I,U}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)}$: We check three things:

- Is π_{att} a valid opening proof of $\mathsf{com}_{\mathsf{att}}$ to att'?
- Is cred a valid signature on $\mathsf{pk}_U || \mathsf{com}_{\mathsf{att}}$ signed by ipk?
- Is σ_{st} a valid signature of st signed by pk_U?

Choice of SPS and Commitment Scheme. From the template above, we can see that the SPS must be able to sign public keys and commitments. Thus, we choose KZG [KZG10] as the commitment scheme due to its short opening proofs and algebraic verification. As for the SPS, many SPS schemes only support Diffie-Hellman message spaces $\mathsf{DH} \subset \mathbb{G}_1 \times \mathbb{G}_2$, which complicates the security argument as the user's public key would then also need to be in DH – which was not the case in the SPS schemes we investigated. Moreover, we also need to support signing of at least two messages. We found that [Gro15] – a unilateral message SPS scheme where the message is in one of the base groups and the public key in the other meets all of our requirements. For the user, we use the same SPS scheme but with the base groups flipped.

Malleability Issues. There is a subtle issue with the above template. If the user only signs st, an adversary can potentially repackage the same signature with different revealed claims/openings. Thereby allowing them to show that the user vouched for the statement st using a completely different set of attributes $\tilde{\mathsf{att}}$. In our full construction, we therefore sign both the statement AND the revealed attributes, thereby binding them together.

Proof-system instantiation for Verify. Our construction instantiates the ShowVRC protocol by decomposing the statement into two parts: an algebraic component and an arithmetic component. The algebraic component—identifying valid credentials and signatures—consists purely of pairing product equations (specifically, checks for the issuer's SPS signature, the user's SPS signature, and the KZG commitment opening). These equations are proven using the Groth-Sahai proof system. The remaining arithmetic constraints (checking the predicate f, PRF evaluations, and nullifier non-revocation) are handled by a commit-and-prove SNARK (e.g., [CFQ19]). We link the two proofs by observing that Groth-Sahai commitments to KZG commitments are compatible with the commit-and-prove interface, ensuring that the algebraic and arithmetic components are consistent with respect to the same underlying attributes.

3 Preliminaries

We will use $\mathbb{G}$ to denote the group. For any positive integer n, we denote by $[n]$ the set $\{1,\ldots,n\}$. For any positive integer k, we denote the k size vector $\vec{x} = (x_1,\ldots,x_k)$.

For an NP language $\mathcal{L}$, the corresponding relation is denoted as $R_{\mathcal{L}}$.

3.1 Bilinear maps

We briefly recall the definition of an asymmetric bilinear map [Jou04, BF01]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be distinct groups, all of prime order q, and let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ onto the target group $\mathbb{G}_T$. Let g_1, g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is an asymmetric bilinear group if the following conditions are met:

- (Efficiency) The group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as well as the mapping $e(\cdot,\cdot)$ are all efficiently computable.
- (Non-degeneracy) $e(g_1, g_2) = g_T$, where g_T is a generator of $\mathbb{G}_T$.
- (Bilinearity) $e(g_1^a, g_2^b) = g_T^{ab}$ for all $a, b \in \mathbb{Z}_q$.

Generic bilinear group model. We will use an extension of the generic group model [Nec94, Sho97] adapted to bilinear groups. We follow the presentation in [BFKT24], in particular defining the label sets as known sets of size equal to the size of the group. Moreover, we define the Add interface to directly support arbitrary linear combinations of group elements, which is without loss of generality due to the double-and-add algorithm.

Definition 1 (Generic Bilinear Group Oracle). *A generic bilinear group oracle is a stateful oracle* GGM*, parameterized by a prime $p \in \mathbb{Z}$, and three sets S_1, S_2, S_T of size p.*

- GGM.Init()*: the oracle samples and stores three uniformly random injections $\tau_i : \mathbb{Z}_p \to S_i$ for $i \in \{1,2,T\}$. Future queries to* GGM.Init *are ignored.*
- GGM.GetGenerator(i)*: the oracle returns $\tau_i(1)$.*
- GGM.Add(i, g_1, g_2, c_1, c_2)*: return $\bot$ unless $g_1, g_2 \in S_i$ and $c_1, c_2 \in \mathbb{Z}_p$. Otherwise, return $g = \tau_i(c_1\tau_i^{-1}(g_1) + c_2\tau_i^{-1}(g_2))$.*
- GGM.Pair(g_1, g_2)*: return $\bot$ unless $g_1 \in S_1, g_2 \in S_2$. Otherwise, return $\tau_T(\tau_1^{-1}(g_1) \cdot \tau_2^{-1}(g_2))$.*

Modeling a group-output random oracle. We will also model the scenario in which parties have access to a cryptographic hash function that outputs group elements, which is needed for example to support BLS signatures [BLS01]. Since the sets S_1, S_2, S_T are public, this is straightforward to implement via a random oracle defined from $\mathbb{Z}_p \to S_i$.

Definition 2. *Let $\boldsymbol{L} = (\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T)$ be three sets of polynomial with elements in $\mathbb{Z}_p[X_1, \ldots, X_n]$. We define the completion of $\boldsymbol{L}$ to be $\boldsymbol{C}(\boldsymbol{L}) = \{\boldsymbol{L}_1 \otimes \boldsymbol{L}_2\} \cup \boldsymbol{L}_T = \{f_i(X_1, \ldots, X_n)\}_i$. A polynomial $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$ is said to be dependent on $\boldsymbol{L} = (\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T)$ if there exist coefficients $\{\kappa_i\}_i \in \mathbb{Z}_p$ such that:*

$$f(X_1, \ldots, X_n) = \sum_{i=1} \kappa_i \cdot g_i(X_1, \ldots, X_D).$$

Otherwise, we say that f is independent of $\boldsymbol{L}$.

3.2 Commitment Schemes

Definition 3 (Non-interactive Bit Commitment Schemes in the CRS Model). *A non-interactive bit commitment scheme in the CRS model is a pair of polynomial-time algorithms* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{com})$ *defined as follows:*

- $\mathsf{Setup}(1^\lambda)$ *outputs a common reference string* $\mathsf{crs} \in \{0,1\}^{\mathsf{poly}(\lambda)}$.
- $\mathsf{com}(\mathsf{crs}, b; r)$ *is a polynomial-time computable function*

$$\mathsf{com} : \{0,1\}^{\mathsf{poly}(\lambda)} \times \{0,1\} \times \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)} \ .$$

The scheme must satisfy the following properties:

Binding: *For all* crs *in the support of* $\mathsf{Setup}(1^\lambda)$*, for any $r, r' \in \{0,1\}^\lambda, b, b' \in \{0,1\}$, if* $\mathsf{com}(\mathsf{crs}, b; r) = \mathsf{com}(\mathsf{crs}, b'; r')$ *then $b = b'$.*

Computational Hiding: *The following holds:*

$$\left\{\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda), \mathsf{com}(\mathsf{crs}, 0; r) : r \leftarrow\!\!\$\ \{0,1\}^\lambda\right\} \approx_c \left\{\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda),\ \mathsf{com}(\mathsf{crs}, 1; r) : r \leftarrow\!\!\$\ \{0,1\}^\lambda\right\} \ ,$$

where computational indistinguishability is with respect to arbitrary non-uniform PPT *distinguisher.*

3.3 Signature Schemes

In this section we define (standard) digital signatures.

Syntax. A digital signature scheme consists of the following polynomial-time algorithms:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$**.** This is a probabilistic algorithm that takes as input the security parameter 1^λ. It outputs a signing key sk and a verification key vk.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m}) \to \sigma$**.** This is a probabilistic algorithm that takes as input the signing key sk and a message $\mathsf{m} \in \{0,1\}^\lambda$. It outputs a signature σ.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma) \to 0/1$**.** This is a deterministic algorithm that takes as input the verification key vk, a message $\mathsf{m} \in \{0,1\}^\lambda$ and a signature σ. It outputs a bit (1 to accept, 0 to reject).

Definition 4. *A digital signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is required to satisfy the following properties:*

Correctness. *For any $\lambda \in \mathbb{N}$ and* $\mathsf{m} \in \{0,1\}^\lambda$,

$$\Pr\left[\ \mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma) = 1 \quad : \quad \begin{array}{l} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m}) \end{array}\ \right] = 1 \ .$$

Unforgeability. *For any admissible poly-size adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\ \mathsf{Verify}(\mathsf{vk}, \mathsf{m}^*, \sigma^*) = 1 \quad : \quad \begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk}) \end{array}\ \right] \le \mathsf{negl}(\lambda)\ ,$$

where we say that $\mathcal{A}$ is admissible if it did not query the $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *oracle with* m^*.

Remark 1 (Message space). *We assume that the message space is $\{0,1\}^\lambda$. This is without loss of generality, since we can sign arbitrary messages by first applying a hash function (that satisfies targeted collision resistance), then signing the hashed message [NY89].*

3.4 Pseudorandom Functions

A function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^n \mapsto \{0,1\}^m$ is a pseudorandom function if for every polynomial time oracle aided distinguisher $\mathcal{D}$, the following holds

$$\left|\Pr_k\left[\ \mathcal{D}^{\mathsf{PRF}(k,\cdot)}(1^\lambda) = 1\ \right] - \Pr_k\left[\ \mathcal{D}^{f(\cdot)}(1^\lambda) = 1\ \right]\right| \le \mathsf{negl}(\lambda)$$

where k is a random string in $\{0,1\}^\lambda$ and f is a function sampled randomly from the set of all function from $\{0,1\}^n$ to $\{0,1\}^m$.

3.5 Non-interactive Zero-knowledge

Definition 5. *A tag-based simulation extractable-non-interactive zero-knowledge (SE-NIZK) proof system* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{SimProve})$ *for an* NP *relation $R_\mathcal{L}$ consists of the polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$*: Takes as input a security parameter, and outputs a common reference string* crs *and trapdoor* td.
- $\mathsf{Prove}(\mathsf{crs}, t, x, w) \to \pi$*: Takes as input* crs*, a tag $t \in \{0,1\}^*$, and any pair $(x, w) \in R_\mathcal{L}$, and outputs a proof π for the statement $x \in \mathcal{L}$ under tag t.*
- $\mathsf{Verify}(\mathsf{crs}, t, x, \pi) \to b$*: Takes as input* crs*, tag t, statement x and proof π and outputs a bit b indicating whether verification has passed or failed.*
- $\mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, t, x) \to \pi$*: Takes as input* crs*, trapdoor* td*, tag t and statement x and outputs a* simulated *proof π under tag t.*

In places where the tag argument is omitted for readability, it is understood to be either fixed globally or encoded into the statement. We will drop the crs *term wherever implicit. A tag-based SE-NIZK satisfies the following properties:*

Perfect Completeness. *A tag-based SE-NIZK satisfies perfect completeness if for any $(x, w) \in \mathcal{R}$ and any tag $t \in \{0,1\}^*$, we have*

$$\Pr\left[\ \mathsf{Verify}(\mathsf{crs}, t, x, \pi) = 1\ :\ \begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, t, x, w) \end{array}\ \right] = 1$$

Proof of knowledge. *For every PPT adversary $\mathcal{A}$, there is a PPT extractor* Ext *such that*

$$\Pr\left[\ \begin{array}{r} \mathsf{Verify}(\mathsf{crs}, t, x, \pi) = 1\ \wedge \\ (x, w) \notin R_\mathcal{L} \end{array}\ :\ \begin{array}{r} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (t, x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(\mathsf{crs}, t, x, \pi) \end{array}\ \right] \le \mathsf{negl}(\lambda)$$

(Multi-theorem) Computational Zero-knowledge. *For every PPT adversary $\mathcal{D}$,*

$$\Big|\Pr\left[\ \mathcal{D}^{\mathsf{O}_{\mathsf{real}}}(\mathsf{crs}) = 1\ :\ (\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda)\ \right] - \Pr\left[\ \mathcal{D}^{\mathsf{O}_{\mathsf{sim}}}(\mathsf{crs}) = 1\ :\ (\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda)\ \right]\Big| \leq \mathsf{negl}(\lambda).$$

where the oracles are defined on inputs (t, x, w) to be $\mathsf{O}_{\mathsf{real}}(t, x, w) = \mathsf{Prove}(\mathsf{crs}, t, x, w)$ for $(x, w) \in R_{\mathcal{L}}$, and $\mathsf{O}_{\mathsf{sim}}(t, x, w) = \mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, t, x)$ for $(x, w) \in R_{\mathcal{L}}$.

Tag-based Weak Simulation-Extractability. *For every PPT adversary $\mathcal{A}$, there exists a PPT extractor* Ext *such that for all $z \in \{0,1\}^*$*

$$\Pr\left[\begin{array}{r} \mathsf{Verify}(\mathsf{crs}, t, x, \pi) = 1\ \wedge \\ (x, w) \notin R_{\mathcal{L}}\ \wedge \\ (t, x) \notin Q \end{array}\ :\ \begin{array}{r} (\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (t, x, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}}(\mathsf{crs}, z) \\ w \leftarrow \mathsf{Ext}(\mathsf{crs}, t, x, \pi, z) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ has oracle access to $\mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, \cdot, \cdot)$*, and Q is the set of tag-statement pairs (t, x) queried by the adversary.*

We will also demand that the proof is straight-line extractable, i.e. there exists an extractor that takes as input the trapdoor td, and works for all adversaries. This means that the extractor can, on input a valid proof extract a witness with overwhelming probability without rewinding the prover.

3.6 Groth-Sahai proofs

Groth and Sahai [GS08] constructed efficient non-interactive zero-knowledge proof systems (Setup, Prove, Verify) for statements that involve equations over bilinear maps. "GS proofs" prove statements that consist of the following types of equations over variables $X_1, \ldots, X_m \in \mathbb{G}_1, Y_1, \ldots, Y_n \in \mathbb{G}_2, x_1, \ldots, x_{m'}, y_1, \ldots, y_{n'} \in \mathbb{Z}_p$.

- *Pairing product equations.* $\prod_{i=1}^{n} e(A_i, Y_i) \prod_{i=1}^{m} e(X_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(A_i, B_j)^{c_{ij}} = 1_T$, for constants $A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, c_{ij} \in \mathbb{Z}_p$, where 1_T is the identity element in $\mathbb{G}_T$. [1]
- *Multi-scalar exponentiation.* $\prod_{i=1}^{n'} A_i^{y_i} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n'} X_i^{c_{ij} y_j} = T_1$, for constants $A_i, T_1 \in \mathbb{G}_1, b_i, c_{ij} \in \mathbb{Z}_p$ and analogous statements for multi-scalar exponentiation in $\mathbb{G}_2$.
- *Quadratic equations in $\mathbb{Z}_p$.* $\sum_{i=1}^{n'} a_i y_i + \sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} c_{ij} x_i y_j \equiv t \bmod p$, for constant $a_i, b_i, c_{ij}, t \in \mathbb{Z}_p$.

GS proofs are in the *common random string* model, and satisfy the completeness and zero-knowledge properties described in Definition 5. However, they only satisfy a weaker notion of knowledge extraction which has been referred to as *partial* knowledge extraction [Gro07]. This property states that if the witness consists of both group elements and exponents, only the group elements are extractable.

Definition 6 (Polynomial Commitments). *A univariate polynomial commitment scheme* pcs *is characterized by the following algorithms:*

- $\mathsf{pcs.Setup}(1^\lambda, 1^d) \to \mathsf{p}$: *The setup algorithm takes a security parameter and a degree bound d as input and outputs a CRS, which is an implicit input to all subsequent algorithms.* p *specifies a finite field $\mathbb{Z}_p$ of prime order p.*
- $\mathsf{pcs.Commit}(f(X) \in \mathbb{Z}_p^{\leq d}[X]) \to (\mathsf{com}_{\mathsf{pcs}}, \mathsf{aux}_{\mathsf{pcs}})$: *The (randomized) commitment algorithm takes a degree $\leq d$ polynomial $f(X)$ as input and outputs a commitment* $\mathsf{com}_{\mathsf{pcs}}$ *and auxiliary information* $\mathsf{aux}_{\mathsf{pcs}}$.

[1] [GS08] also consider pairing product equations where the target element is not the identity, but their proofs for such equations are in general only witness indistinguishable, as opposed to zero-knowledge.

- $\mathsf{pcs.Open}(\mathsf{com_{pcs}}, \mathsf{aux_{pcs}}, x \in \mathbb{Z}_p) \rightarrow (y \in \mathbb{Z}_p, \pi_{\mathsf{pcs}})$: *The (randomized) opening algorithm takes* $\mathsf{com_{pcs}}$, $\mathsf{aux_{pcs}}$ *and an opening point* $x \in \mathbb{Z}_p$ *as input and outputs the opening value* $y \in \mathbb{Z}_p$ *and a proof* π_{pcs}.
- $\mathsf{pcs.Verify}(\mathsf{com_{pcs}}, x, y, \pi_{\mathsf{pcs}}) \rightarrow 0/1$: *The (deterministic) verification algorithm takes a commitment* $\mathsf{com_{pcs}}$*, an opening point* x*, a claimed value* y*, and an opening proof* π_{pcs}*; it outputs* 1 *if* π_{pcs} *verifies with respect to* $\mathsf{com_{pcs}}$*,* x*, and* y*.*

A secure pcs *satisfies correctness, binding, evaluation-binding and hiding. We additionally require the* pcs *to be extractable as defined below.*

PC Correctness: *For all* $\lambda \in \mathbb{N}$*,* $d = \mathrm{poly}(\lambda)$*, polynomials* $f(X) \in \mathbb{Z}_p^{\leq d}[X]$*, and points* $x \in \mathbb{Z}_p$*, it holds that*

$$\Pr\left[\mathsf{pcs.Verify}(\mathsf{com_{pcs}}, x, y, \pi_{\mathsf{pcs}}) = 1 \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{pcs.Setup}(1^\lambda, 1^d) \\ (\mathsf{com_{pcs}}, \mathsf{aux_{pcs}}) \leftarrow \mathsf{pcs.Commit}(f(X)) \\ (y, \pi_{\mathsf{pcs}}) \leftarrow \mathsf{pcs.Open}(\mathsf{com_{pcs}}, \mathsf{aux_{pcs}}, x) \end{array}\right] > 1 - \mathsf{negl}(\lambda)$$

PC Binding: *For all* $\lambda \in \mathbb{N}$*,* $d = \mathrm{poly}(\lambda)$*, and PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{r} \mathsf{pcs.Commit}(f(X); r) = \mathsf{pcs.Commit}(f'(X); r') \\ \wedge\, f(X) \neq f'(X) \end{array} \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{pcs.Setup}(1^\lambda, 1^d) \\ (f(X), f'(X), r, r') \leftarrow \mathcal{A}(\mathsf{p}) \end{array}\right] = \mathsf{negl}(\lambda)$$

PC Evaluation-Binding: *For all* $\lambda \in \mathbb{N}$*,* $d = \mathrm{poly}(\lambda)$*, and PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{r} \mathsf{pcs.Verify}(\mathsf{com_{pcs}}, x, y', \pi'_{\mathsf{pcs}}) = 1 \\ \wedge\, \mathsf{pcs.Verify}(\mathsf{com_{pcs}}, x, y, \pi_{\mathsf{pcs}}) = 1 \\ \wedge\, y \neq y' \end{array} \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{pcs.Setup}(1^\lambda, 1^d) \\ (\mathsf{com_{pcs}}, x, y, y', \pi_{\mathsf{pcs}}, \pi'_{\mathsf{pcs}}) \leftarrow \mathcal{A}(\mathsf{p}) \end{array}\right] = \mathsf{negl}(\lambda)$$

PC Statistical (Computational) Hiding: *For all* $\lambda \in \mathbb{N}$*,* $d = \mathrm{poly}(\lambda)$*, subsets* $S \subset \mathbb{Z}_p$ *of size* $\leq d$*, polynomials* $f_0(X), f_1(X) \in \mathbb{Z}_p^{\leq d}[X]$ *such that* $f_0(x) = f_1(x)$ *for all* $x \in S$*, and unbounded (PPT) adversaries* $\mathcal{A}$*,*

$$\Pr\left[\mathcal{A}(\mathsf{p}, \mathsf{com}_b, (f_b(x), \pi_x^b)_{x \in S}) = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{p} \leftarrow \mathsf{pcs.Setup}(1^\lambda, 1^d) \\ (\mathsf{com}_0, \mathsf{aux}_0) \leftarrow \mathsf{pcs.Commit}(f_0(X)) \\ \{(f_0(x), \pi_x^0) \leftarrow \mathsf{pcs.Open}(\mathsf{com}_0, \mathsf{aux}_0, x)\}_{x \in S} \\ (\mathsf{com}_1, \mathsf{aux}_1) \leftarrow \mathsf{pcs.Commit}(f_1(X)) \\ \{(f_1(x), \pi_x^1) \leftarrow \mathsf{pcs.Open}(\mathsf{com}_1, \mathsf{aux}_1, x)\}_{x \in S} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

PC Extractability: *The opening and verification algorithms of the PCS scheme constitute a non-interactive proof system for the language*

$$\left\{(\mathsf{com_{pcs}}, x, y) \;\middle|\; \exists\, (f(X), r) \in \mathbb{Z}_p^{\leq d} \times \{0,1\}^* : \mathsf{pcs.Commit}(f(X); r) = \mathsf{com_{pcs}} \wedge f(x) = y\right\}$$

that satisfies knowledge soundness. Note that a PCS scheme satisfies evaluation-binding if it is binding and extractable.

3.7 Hiding KZG Polynomial Commitments

We recall the hiding and extractable variant of the KZG polynomial commitment scheme [KZG10, ZGK+17, KT24].

- $\mathsf{Setup}(1^\lambda, 1^d) \rightarrow \mathsf{p}$: The setup algorithm takes a security parameter λ and a degree bound d as input, generates a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, samples $\tau, \eta \leftarrow \mathbb{Z}_p$, and outputs the following as public parameters:
$$\left(g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}, g^\eta, h^\tau, h^\eta\right) \in (\mathbb{G}_1)^{d+1} \times (\mathbb{G}_2)^2$$

- $\mathsf{Commit}(f(X)) \to (\mathsf{com}, \mathsf{aux})$: The commitment algorithm takes a polynomial $f(X) = \sum_{j=0}^{\deg(f)} f_j X^j$, with $\deg(f) \leq d$ as input, samples a random integer $r \leftarrow \mathbb{Z}_p$, and outputs $\mathsf{com} = g^{f(\tau)+\eta r}$ and aux computed as follows:

$$\mathsf{com} = (g^\eta)^r \cdot \prod_{j=0}^{\deg(f)} (g^{\tau^j})^{f_j} \qquad \mathsf{aux} = (f(X), r)$$

- $\mathsf{Open}(\mathsf{com}, \mathsf{aux}, x) \to (f(x), \pi)$: The opening algorithm takes com, and $\mathsf{aux} = (f(X), r)$ as well as an opening point $x \in \mathbb{Z}_p$ as input, samples a random integer $s \leftarrow \mathbb{Z}_p$, and outputs the opening proof

$$\pi = \left((g^\eta)^s \cdot g^{f_x(\tau)},\ g^{(r+sx)}/(g^\tau)^s\right),$$

where $f_x(X) = \frac{f(X)-f(x)}{X-x}$ is the quotient polynomial, and $g^{f_x(\tau)}$ is computed as shown above using the polynomial $f_x(X)$.

- $\mathsf{Verify}(\mathsf{com}, x, y, \pi) \to 0/1$: The verification algorithm takes a commitment com, an opening point $x \in \mathbb{Z}_p$, a claimed value $y \in \mathbb{Z}_p$, and an opening proof $\pi = (\pi^1, \pi^2) \in (\mathbb{G})^2$. It outputs 1 iff

$$e(\mathsf{com}/g^y, h) = e(\pi^1, h^\tau/h^x) \cdot e(\pi^2, h^\eta)$$

- $\mathsf{BatchOpen}(\mathsf{com}, \mathsf{aux}, (x_i)_{i\in[k]}) \to \left((f(x_i))_{i\in[k]}, \pi\right)$: The batch opening algorithm takes com, $\mathsf{aux} = (f(X), r)$ and a sequence of opening points $(x_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$ as input, samples a random integer $s \leftarrow \mathbb{Z}_p$, and finds the degree-$(k-1)$ polynomial $r(X)$ such that $r(x_i) = f(x_i)$ for $i \in [k]$, and the degree-k polynomial $z(x) = \prod_{i\in[k]}(X - x_i)$. It then outputs the opening proof

$$\pi = \left((g^\eta)^s \cdot g^{f_{(x_i)_{i\in[k]}}(\tau)}, g^r/(g^{z(\tau)})^s\right),$$

where $f_{(x_i)_{i\in[k]}}(X) = \frac{f(X)-r(X)}{\prod_{i\in[k]}(X-x_i)}$ is the quotient polynomial.

- $\mathsf{BatchVerify}(\mathsf{com}, (x_i)_{i\in[k]}, (y_i)_{i\in[k]}, \pi) \to 0/1$: The verification algorithm takes a commitment com, a sequence of opening points $(x_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$, claimed values $(y_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$, and an opening proof $\pi = (\pi^1, \pi^2)$. It outputs 1 iff

$$e(\mathsf{com}/g^{r(\tau)}, h) = e(\pi^1, h^{z(\tau)}) \cdot e(\pi^2, h^\eta)$$

Before we state the security guarantees of the scheme, we recap the q-Discrete-Logarithm assumption.

Definition 7. *The q-Discrete-Logarithm assumption, denoted by q-DLOG and parameterized by $q \in \mathbb{N}$, states that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\tau' = \tau \;\middle|\; \tau \leftarrow \mathbb{Z}_p,\ \tau' \leftarrow \mathcal{A}(g, g^\tau, \ldots, g^{\tau^q}, h, h^\tau, \ldots, h^{\tau^q})\right] = \mathsf{negl}(\lambda)$$

Extractability of the PCS scheme is shown in the Algebraic Group Model [FKL18], which assumes that any adversary that outputs a group element also outputs its representation as a linear combination of the group elements previously received. The following was proven in [KT24, Sections 3.5.3, B.2.2].

Theorem 4. *The PCS scheme above satisfies completeness, statistical hiding, binding under the d-DLOG assumption, and extractability under the $(d+1)$-DLOG assumption in the Algebraic Group Model.*

4 Personhood Oracle

In this section, we formally model the ability of adversarial users to obtain credentials of personhood by deceiving the issuing authority (issuer). Typically, when a credential is issued, the user requests credentials for a set of attributes. Whether these attributes truly "match" the user is often considered out of scope, modeled simply by a catch-all $\perp$ symbol output by the issuer to indicate a refusal to issue a credential.

However, by explicitly modeling the adversarial ability to *fool* the issuer, our goal is twofold: (i) to capture more realistic scenarios that occur in practice; and (ii) to understand the guarantees provided in these settings.

We describe below the various components including the personhood attribute validation (PAV) oracle O.

Users and attributes. We consider a set of *users* $\{U_1, \ldots, U_m\}$, each user U_i has as input a set of attributes $\mathsf{att}_i \in \{0,1\}^*$. These are provided to the user by the external environment $\mathcal{E}$. An honest user will always use the provided attributes, whereas an adversarial user may choose to use any attributes.

Issuers. In addition to the aforementioned users, we will also have special parties called *issuers* $\{I^{(i)}, \ldots, I^{(n)}\}$. The issuers interact with the user to provide credentials to the user.

Truthfulness of attributes. To establish whether a user attribute is *true*, we define a predicate $\mathbb{T}$,

$$\mathbb{T} : \mathscr{I} \times \{0,1\}^* \mapsto \{0,1\}.$$

In addition to taking as input $\mathsf{att} \in \{0,1\}^*$, it also takes an abstract input we denote by $\mathscr{U}$ from the input space $\mathscr{I}$. We interpret $\mathscr{U}$ as encoding all external ground truth relevant to the attribute validation process (including the identity of the user being validated). Importantly, we *do not* require this function to be efficiently computable. See Section 4.1 for further discussion.

Personhood Attribute Validation (PAV) oracle. We now describe the oracle that is used by the user and issuer to validate attributes presented by a user. The user sends its attributes to the issuer, and provide, respectively, as inputs to the oracle the abstract input and attributes. The oracle returns either 0 or 1 to the issuer to indicate whether it considers the attributes valid. To capture that this validation process is sometimes faulty, we allow for the scenario that when the presented attributes are not valid with respect to the abstract input provided by the user (as defined by the predicate $\mathbb{T}$), the oracle may still incorrectly return 1. However, to obtain any meaningful guarantees one must bound this error.

Further, each issuer may have its own validation process, and therefore allow a separate oracle for each issuer. We describe this formally below by specifying the properties the oracle must satisfy.

Let $\mathcal{D}^{(i)} = \{\mathcal{D}^{(i)}_\lambda\}_\lambda$ be a distribution family for an issuer I_i, where each $\mathcal{D}^{(i)}_\lambda$ samples a deterministic oracle O_λ of the form,

$$O^{(i)}_\lambda : \mathscr{I} \times \{0,1\}^{m(\lambda)} \to \{0,1\}$$

for some polynomial m.

Definition 8. *We say that the oracle ensemble $\mathcal{D}^{(i)} = \{\mathcal{D}^{(i)}_\lambda\}_\lambda$ has one-sided $\varepsilon(\lambda)$-error if it satisfies the following properties:*

Completeness *For every $\lambda \in \mathbb{N}$, $x \in \{0,1\}^{m(\lambda)}$, $\mathscr{U} \in \mathscr{I}$ where $\mathbb{T}(\mathscr{U}, x) = 1$, we have*

$$\Pr_{O^{(i)}_\lambda \leftarrow \mathcal{D}^{(i)}_\lambda}\left[\ O_\lambda(\mathscr{U}, x) = 1\ \right] \geq 1 - \mathsf{negl}(\lambda).$$

One-sided error. *For every $\lambda \in \mathbb{N}$, $x \in \{0,1\}^{m(\lambda)}$, $\mathscr{U} \in \mathscr{I}$ where $\mathbb{T}(\mathscr{U}, x) = 0$, we have*

$$\Pr_{O^{(i)}_\lambda \leftarrow \mathcal{D}^{(i)}_\lambda}\left[\ O_\lambda(\mathscr{U}, x) = 1\ \right] \leq \varepsilon(s).$$

We will later interpret s as the adversarial spend against the issuer.

As stated previously, each issuer will have its own oracle. This in turn means that each issuer can have its own ε. We would like to emphasize that the ε is not a static parameter. In particular, instead of universal ε, we want to model the fact that a "more powerful" adversary can increase ε. In the context of attribute validation, we think of the financial spending by an adversary (to be defined shortly) to have an impact on ε. Thus, to formalize this we define ε to be a *non-decreasing* function

$$\varepsilon : \mathbb{N} \to [0, 1],$$

where the input is the spend $s \in \mathbb{N}$. As discussed previously, each issuer will have a different function $\varepsilon^{(i)}$. Note that the above description is oblivious to the adversarial strategy, and is only a function of the spending s. We now describe the adversary's spending below.

Adversary spend. To model the spending of an adversary, we consider the adversary to have some budget B (potentially provided by the environment $\mathcal{E}$) that it can choose to split as spending between the various issuers. Formally, for a set of n issuers, we define for adversary $\mathcal{A}$ a *budget* function:

$$\mathsf{Budget}_{\mathcal{A}} : \mathbb{N} \mapsto \mathbb{N}^n,$$

with the requirement that budget is conserved, i.e. for all budget $B \in \mathbb{N}$,

$$B = \sum_{i=1}^{n} \mathsf{Budget}_{\mathcal{A},i}(B),$$

where $\mathsf{Budget}_{\mathcal{A},i}(B)$ is the spending for the i-th issuer on input budget B.

Remark 2 (Other imperfect validation oracles). *The PAV oracle models the fact that attribute validation by issuers may be imperfect. One could analogously define a* statement validation (SV) oracle *governing the issuance of relationship attestations between users: such an oracle would take as input a statement* st *and return a bit indicating whether the statement is considered valid, with a similar one-sided error guarantee.*

In this work we focus on PAV oracles for clarity, but the model extends naturally to additional imperfect validation oracles for other interactions. Incorporating such oracles would allow one to reason about the reliability of attestations and other statements in exactly the same way as we analyze credential issuance, and would yield analogous confidence guarantees for shows that depend on those statements.

4.1 Discussion

We discuss below some choices made in the modeling of the personhood attribute verification oracles.

Abstract user input. As stated previously, we use the abstract user input $\mathcal{U}$ to model the ground truth of a user, outside of external or adversarial control. Since the function $\mathbb{T}$ is not required to be efficiently computable, nor do we require this input $\mathcal{U}$ to interact with the system outside of the oracle, we keep the input and the input space abstract. However we do want to emphasize that the input $\mathcal{U}$ is *not* static. For instance, one can imagine the input to change over time. Since modeling time, and the evolution of this abstract input, is almost totally orthogonal to this work we choose the above simpler formulation. We feel that defining epochs, allowing the adversary to update the clock, and (potentially) changing $\mathbb{T}$ at every epoch would add complexity to the paper without any useful insights. That being said, if a future formalization finds this useful, it would be simple enough to modify our construction to formally model time.

Correlated fooling success. In our description of the model, we specify each oracle to be *independently* sampled, and thus the success of a cheating adversary for each oracle is independent. This allows for a cleaner analysis. However, one can consider more complicated correlations, where the success of an adversary with respect to the oracle of an issuer is correlated with its success with the oracle of a different issuer. While the analysis in the simpler independence setting can be extended to the correlated setting, albeit in a more tedious manner.

Different Security Levels. In our theorem statement, we will state our results with respect to a single security parameter. In practice, each issuer may have its own security parameter, i.e. the security parameter for issuer I is $\lambda^{(I)}$. Again, our choice of a single security parameter is motivated by a clean model. If the simplest setting, the security parameters are polynomially related, and the results extend in a straightforward manner. Any changes by a single issuer to their security parameter corresponds to changing their " security polynomial". However one could also consider writing out the theorem explicitly in terms of the various security parameters in scenarios where this is important.

5 Definition

We present the security guarantees of the proof-of-personhood (PoP) functionality using the real/ideal world framework. Specifically, we define an ideal functionality $\mathcal{F}_{\mathsf{PoP}}$ that captures the desired security properties. We refer to this as the *base* functionality. Further, we support additional privacy guarantees for the users in the form of unlinkability. To illustrate the difference from the base functionality, the changes to the functionality and the surrounding discussion are highlighted in a box. For instance, the ideal functionality is now defined to be $\mathcal{F}_{\mathsf{PoP}\text{-user-unlinkability}}$ with the changes for the privacy guarantees clearly marked.

Looking ahead, we provide constructions for both $\mathcal{F}_{\mathsf{PoP}}$ and $\mathcal{F}_{\mathsf{PoP}\text{-user-unlinkability}}$, with the construction satisfying the former being more efficient. In systems where strong privacy guarantees are not required, one may consider using a protocol satisfying only $\mathcal{F}_{\mathsf{PoP}}$.

Before providing its formal specification, we describe its components below.

Participants. There are two types of parties that interact with the functionality: *users* and *credential issuers*. A single party may act in both roles. The identity of each credential issuer is assumed to be known to all users. We consider a *static adversary* that can corrupt any subset of users and credential issuers, who may freely collude. Users may additionally act as attestation issuers (issuing VRCs). Unlike credential issuers, attestation issuers need not have publicly known identities.

State of the ideal functionality. The functionality maintains two lists: a list of credentials S_C and a list of relationship attestations S_A, which are updated as participants interact with the functionality to record all issued credentials and attestations.

Contexts. For attestation issuers, we define a context for each attestation. Looking ahead, users can be identified separately within each context, but the same user providing attestations across contexts cannot be linked. See Section 5.3 for further discussion on our choice of using contexts.

Interaction with the ideal functionality. Parties interact with $\mathcal{F}_{\mathsf{PoP}}$ and $\mathcal{F}_{\mathsf{PoP}\text{-user-unlinkability}}$ as follows, with changes to the interaction with $\mathcal{F}_{\mathsf{PoP}\text{-user-unlinkability}}$ highlighted accordingly:

- *Credential issuance:* To issue a credential, a user submits a request specifying the desired issuer. The functionality forwards the request to the issuer and updates S_C if the issuer approves.
- *Relationship attestation issuance:* To obtain an attestation, a user specifies the target user and the statement to be attested. The functionality relays the request and updates S_A if the specified user approves.
- *Context key issuance*: A user can request the generation of a context dependent key.
- *Showing attestations:* To show an attestation, a user specifies a function, list of attestations and a recipient. The functionality computes the function on the specified attestations and sends the result to the recipient.

Unlinkability during show. Our ideal functionality will provide different unlinkability guarantees depending on whether the functionality is $\mathcal{F}_{\mathsf{PoP}}$ or $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$. For the base functionality, $\mathcal{F}_{\mathsf{PoP}}$ we have the following unlinkability:

Show-attestation unlinkability: When a showing of attestation is requested for a function f, the ideal functionality provides the function f to the recipient if f output 1, thereby hiding the underlying attestations used in the computation of f's output and providing unlinkability between the attestations and showing.

To provide stronger unlinkability for the guarantees from colluding parties that want to link users across multiple interactions, the ideal functionality will sample random ephemeral keys. A mapping between the users and the ephemeral keys are stored, with the functionality *only* revealing the ephemeral keys to the participants. We specify below the different types of unlinkability guarantees provided.

Credential receiver unlinkability: Colluding issuers should not be able to link a user across multiple credential requests. When a user requests a credential, the functionality samples a random string, which it sends to the issuer as the *ephemeral key* of the requesting user. The functionality records in S_C the mapping between the user's long-term identity and the sampled key.

Attestation issuer cross-context unlinkability: An attestation receiver should not be able to link an attestation issuer between contexts, but linking within a context is permitted. A user can request a context specific key, and the ideal functionality samples an ephemeral key for the context, storing the mapping.

Attestation receiver unlinkability: Colluding attestation issuers should not be able to map attestation receivers across multiple interactions. Similar to the credential request, the ideal functionality samples an ephemeral key when a user requests an attestation. This ephemeral key is unique with respect to the context specific key that the user is requesting an attestation from.

We emphasize that unlinkability is not required for showing. In particular, the recipient of a show request may learn the long-term identity of the requester, and multiple showings by the same user may be linkable. Our unlinkability guarantees apply only to credential and attestation issuance phases.

Remark 3. *We note that one could conceive of "intermediate" security notions where not all of the unlinkability properties are achieved.*

5.1 Security

We now describe what it means to securely realize the ideal functionality, in the presence of personhood attribute validation (PAV) oracles.

Protocol execution in the ideal world. In the ideal world, parties interact with the ideal functionality $\mathcal{F}_{\mathsf{PoP}}$ (or $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$) and additionally have oracle access to the issuer-specific PAV oracles defined in Section 4. For each issuer $I^{(i)}$, an oracle $\mathcal{O}_\lambda^{(i)}$ is sampled once from the distribution $\mathcal{D}_\lambda^{(i)}$ at setup. These oracles are treated as global objects available identically in the real and ideal worlds and the simulator is given the same oracle access as the parties. The ideal functionality itself never queries these oracles and never receives $\mathscr{U}$.

When the environment $\mathcal{E}$ instructs an honest user U to request a credential from issuer $I^{(i)}$ with attributes att, the user sends $(\mathsf{credIssuance}, I^{(i)}, \mathsf{att})$ to the ideal functionality. The functionality forwards the request (or its unlinkable form) to issuer $I^{(i)}$, revealing the attributes att to the issuer.

To determine whether the credential should be issued, the user and issuer make a joint query to the shared oracle $\mathcal{O}_\lambda^{(i)}$: the user provides its abstract input $\mathscr{U}$, and the issuer provides the attributes $\mathsf{att}|_{\mathsf{idx}}$ it received from the functionality. The oracle returns a bit to the issuer, and the issuer approves the credential if and only if the oracle returns 1. Thus the ideal functionality records a credential in S_C exactly when it receives $(\mathsf{credApproved}, \cdot)$ from $I^{(i)}$.

$\mathcal{F}_{\mathsf{PoP}\boxed{\text{-user-unlinkability}}}$

Parties: Credential issuers $\mathcal{I} = \{I_1, \ldots, I_n\}$, a set of users $\{U_1, \ldots, U_m\}$. An adversary $\mathcal{A}$ corrupts a set of parties C, which may additionally include issuers.
Setup: Initialize the set of credentials S_C and relationship attestations S_A to empty sets, i.e. $S_C \leftarrow \emptyset$ and $S_A \leftarrow \emptyset$.
Credential Issuance: On input $(\mathsf{credIssuance}, I, \mathsf{att})$ from a user U where $I \in \mathcal{I}$.

1. $\boxed{\text{The functionality samples a random string } \xi \text{ as the user's pseudonym.}}$
2. The functionality sends to I the tuple $(\mathsf{credIssuance}, \boxed{U}, \boxed{\xi}, \mathsf{att})$.
3. If I responds with $(\mathsf{credApproved}, \boxed{U}, \boxed{\xi})$, add $(U, \boxed{\xi}, \mathsf{att}, I, \boxed{\mathsf{ctxKeys}})$ to S_C, $\boxed{\text{where } \mathsf{ctxKeys} = \{\}}$. Then send $(\mathsf{credApproved}, \boxed{\xi})$ to U.

Context Key Request: On input $(\mathsf{ctxtKey}, \xi, \mathsf{ctxt})$ from a user U,

1. Check there is not already a key corresponding to ctxt in ctxKeys where $(U, \xi, \_, \_, \mathsf{ctxKeys}) \in S_C$. Else, sample a random key η and add (ctxt, η) to ctxKeys and update S_C.

Relationship Attestation Issuance: On input $(\mathsf{relAttestIssuance}, \boxed{V}, \boxed{\eta, \mathsf{ctxt}}, \mathsf{st})$ from a user U,

1. $\boxed{\text{The functionality samples a random string } \xi \text{ as the user } U\text{'s pseudonym.}}$
2. $\boxed{\text{Looks up the user } V \text{ such that } (\mathsf{ctxt}, \eta) \text{ is in ctxKeys for } (V, \_, \_, \_, \mathsf{ctxKeys}).}$
3. Sends to V the tuple $(\mathsf{relAttestIssuance}, \boxed{U}, \boxed{\xi}, \mathsf{st})$.
4. If V responds with $(\mathsf{approveRelAttest}, \boxed{U}, \boxed{\xi}, \mathsf{att}', \mathsf{idx}, I')$, check whether there exists $(V, \_, \mathsf{att}, I', \_) \in S_C$ such that $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$. If such a tuple exists, return $(\mathsf{relAttestApproved}, \boxed{\xi}, \mathsf{att}', \mathsf{idx}, I', \mathsf{id})$ to U and record $(\mathsf{id}, U, \boxed{\xi}, V, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I')$ in S_A, where id is a fresh random identifier.

Showing Attestations: On input $(\mathsf{showCred}, f, \mathsf{ID}, V)$ from user U, the functionality verifies that none of the credentials corresponding to ID are revoked (by consulting an external revocation list). If so, it checks that

$$f(\mathsf{att}_1, \mathsf{st}_1, \eta_1 \cdots, \mathsf{att}_N, \mathsf{st}_N, \eta_N) = 1 \text{ where } (\mathsf{ID}[i], U, \eta_i, \_, \mathsf{st}_i, \mathsf{att}_i, \_, I_i) \in S_A$$

If this holds, it sends $(\mathsf{showCred}, U, f, (\mathsf{st}_1, I_1, \ldots, \mathsf{st}_N, I_N))$ to V; otherwise, it returns $\perp$.

Figure 1: Ideal Functionality for a Proof of Personhood. Options that are applicable only to $\mathcal{F}_{\mathsf{PoP}}$ are represented in $\boxed{\text{dashed boxes}}$, and options applicable only to $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ are represented in $\boxed{\text{solid boxes}}$.

If either party is corrupted, the adversary may deviate arbitrarily and need not follow the oracle output. Apart from this oracle-mediated issuance step, all other interactions proceed exactly as specified by the ideal functionality.

Real-world execution. A proof-of-personhood system Proof of Personhood is specified by a tuple (Setup, IssuePHC, IssueVRC, VerifyVRC, ShowVRC, VerifyShow)

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^t) \rightarrow (\mathsf{pp}, \{\mathsf{isk}^{(i)}\}_{i \in [t]})$: A randomized initialization algorithm that takes as input the security parameter, a parameter ℓ specifying the upper bound on the number of attributes, and a parameter 1^t for the number of credential schemes supported. It outputs public parameters p and issuer secret keys $\{\mathsf{isk}^{(i)}\}_{i \in [t]})$ where issuer $I^{(i)}$ is given $\mathsf{isk}^{(i)}$.

$\mathsf{IssuePHC}^{O_\lambda^{(i)}}(\langle I^{(i)}(\mathsf{pp}, \mathsf{isk}^{(i)}), U(\mathsf{pp}, \mathsf{ipk}^{(i)}, \mathsf{att}_U, \mathsf{pk}_U, \mathsf{sk}_U)\rangle) \rightarrow \mathsf{cred}_U^{(i)}/\perp$: An oracle-aided interactive protocol between a personhood credential issuer $I^{(i)}$ and user U. The issuer has input public parameters pp, its secret issuer key $\mathsf{isk}^{(i)}$ and the user has as input public parameters pp, the issuer public key $\mathsf{ipk}^{(i)}$ a set of ℓ attributes att_U, and a public key and secret key pair $(\mathsf{pk}_U, \mathsf{sk}_U)$. At the end of the protocol, the user gets as output a *personhood* credential $\mathsf{cred}_U^{(i)}$ or the issuance fails with $\perp$.

$\mathsf{IssueVRC}(\mathsf{pp}, \mathsf{cred}_U^{(i)}, \mathsf{att}_U, \mathsf{idx}, \mathsf{pk}_{U'}, \mathsf{st}) \to (\pi, \mathsf{st}, \mathsf{idx}, \mathsf{att}')$: A randomized verifiable relationship credential (VRC) issuance algorithm which takes as input public parameters pp, a personhood credential $\mathsf{cred}_U^{(i)}$, a set of ℓ attributes att, a set of distinct indices $\mathsf{idx} \in [\ell]^\ell$, public key of intended recipient $\mathsf{pk}_{U'}$ and the statement st. The output of the algorithm is a VRC π along with the statement st, indices idx and attributes att'.

$\mathsf{VerifyVRC}(\mathsf{pp}, \mathsf{ipk}^{(i)}, \mathsf{pk}_{U'}, \mathsf{st}, \mathsf{pk}_U, \mathsf{att}, \mathsf{idx}, \pi_U^{(i)}) \to 0/1$: A deterministic algorithm checks if a VRC is valid. It takes as input public parameters pp, an issuer public key $\mathsf{ipk}^{(i)}$, designated user public key $\mathsf{pk}_{U'}$, statement st, VRC issuer public key pk_U, revealed attributes att, indices corresponding to revealed attributes idx, and VRC proof $\pi_U^{(i)}$. The algorithm outputs 0 or 1 to indicate whether it rejects or accepts the proof.

$\mathsf{ShowVRC}(\mathsf{pp}, f, \mathsf{pk}_U, \{\mathsf{ipk}^{(i_j)}, \mathsf{st}_j, \mathsf{pk}_j, \mathsf{att}_j, \mathsf{idx}_j, \pi_j\}_{j \in [N]}) \to (\pi, \mathsf{x})$: A randomized algorithm that generates a VRC show proof. It takes as input public parameters pp, a predicate f, public key of the user pk_U and N additional inputs that include the issuer public key $\mathsf{ipk}^{(i_j)}$, statement st_j, VRC issuer public key pk_j, revealed attibutes att_j, indices corresponding to revealed attributes idx_j, and VRC proof π_j. The output of the algorithm is a pair consisting of the VRC show proof π and show statement x.

$\mathsf{VerifyShow}(\mathsf{pp}, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N), \pi) \to 0/1$: A deterministic algorithm checks if a show is valid. It takes as input public parameters pp, a predicate f, N pairs of statements and indices, and a VRC show proof π. The algorithm outputs 0 or 1 to indicate whether it rejects or accepts the proof.

In the real world, parties execute these algorithms according to the system specification. The environment $\mathcal{E}$ provides inputs to the parties and receives all outputs. An adversary $\mathcal{A}$ controls all corrupted parties and may cause them to deviate arbitrarily from the specified algorithms.

All parties additionally have access to the issuer-specific PAV oracle ensemble from Section 4. For each issuer I, an oracle $O_\lambda^{(I)}$ is sampled once from $\mathcal{D}_\lambda^{(I)}$ and is available identically in both the real and ideal worlds. During credential issuance, the user and issuer jointly query this oracle: the user provides the abstract input and the issuer provides the attributes under consideration.

Security. We say that a proof-of-personhood system Proof of Personhood securely realizes the ideal functionality $\mathcal{F}_{\mathsf{PoP}}$ (respectively $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$) if for every real-world adversary $\mathcal{A}$ there exists an ideal-world simulator $\mathcal{S}$ such that for every environment $\mathcal{E}$, the real and ideal executions are computationally indistinguishable.

- In the real execution, $\mathcal{E}$ interacts with parties running the algorithms of Proof of Personhood, the adversary $\mathcal{A}$, and the PAV oracle ensemble.
- In the ideal execution, $\mathcal{E}$ interacts with the ideal functionality $\mathcal{F}_{\mathsf{PoP}}$ (respectively $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$), the simulator $\mathcal{S}$, and the same PAV oracle ensemble.

The simulator controls all corrupted parties in the ideal world and has the same oracle access as the parties in the real world. The environment outputs a bit at the end of the execution; we require that the difference in acceptance probabilities between the real and ideal worlds is negligible in the security parameter.

5.2 Interpreting the Ideal Functionality

The real/ideal definition ensures that any real-world execution can be simulated in the ideal world with access to the same PAV oracles. However, to understand what assurance a verifier obtains from a successful *show*, it suffices to analyze the ideal-world execution itself.

In the ideal world, each credential issued by an honest issuer is approved only through its PAV oracle. If the attributes presented at issuance are truthful, approval occurs except with negligible probability; if the attributes are false, an honest issuer may still approve only with probability bounded by its one-sided error parameter $\varepsilon^{(i)}(s_i)$, where s_i is the adversarial spend directed at that issuer. Since each issuer samples its oracle independently, approval errors across honest issuers are independent conditioned on the adversary's spending allocation.

Interpreting security in the ideal world. A verifier should not expect any integrity guarantee from credentials issued by corrupted issuers, since a corrupted issuer may approve arbitrary attributes. Accordingly, in the presence of issuer corruptions, the meaningful guarantee is about the subset of credentials in a show that originate from *honest* issuers: these credentials are mis-issued only when the corresponding PAV oracle errs during issuance. We capture this by defining a bad event BAD that ignores credentials from corrupted issuers, and bound its probability below.

Theorem 5 (Ideal-World Mis-Issuance Bound for Honest Issuers). *Fix a security parameter λ and an adversary $\mathcal{A}$ with total budget B, and let $s_i := \mathsf{Budget}_{\mathcal{A},i}(B)$ be the spend allocated to issuer $I^{(i)}$.*

Consider any ideal-world execution in which a verifier accepts a show proof that depends on credentials originating from a set of distinct issuers $\mathcal{I}$. Let BAD *be the event that* at least one credential used in the show that originates from an honest issuer *was mis-issued, i.e., there exists an honest issuer $I^{(i)} \notin C$ and a credential used in the show issued by $I^{(i)}$ on attributes* att *such that $\mathbb{T}(\mathscr{U}, \mathsf{att}) = 0$ at the time of issuance, yet the issuer approved issuance.*

Then, except with negligible probability,

$$\Pr\left[\ \mathsf{BAD}\ \right] \leq 1 - \prod_{i \in \mathcal{I} \setminus C} \left(1 - \varepsilon^{(i)}(s_i)\right),$$

i.e., conditioned on the show being accepted, the probability of mis-issuance decreases multiplicatively, and equivalently the verifier's confidence increases multiplicatively with the number of honest issuers involved.

5.3 Impossibility of Full Unlinkability

We now provide a justification of our choice of defining contexts, and the notion of *attestation issuer cross-context unlinkability*, rather than the more natural requirement of *attestation issuer unlinkability*. This tension arises from the class of functions we wish to support during the *show* phase. Specifically, we describe a predicate function f whose security guarantees are rendered impossible if the system simultaneously achieves *attestation issuer unlinkability*, which we henceforth call *full unlinkability*.

Full unlinkability. Full unlinkability requires that pseudonyms issued to the same user are computationally unlinkable to each other and to the underlying user by any other party, including attestation receivers and show verifiers. To show that such a notion is impossible, we must formally specify the requirements. We describe the functionality $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$ below, by specifying the difference from $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$:

1. $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$ does not support ctxKeys; instead it maintains, for each credential holder, a set Pseudonyms. And stores during credential issuance $(U, \xi, \mathsf{att}, \mathsf{idx}, I, \mathsf{Pseudonyms})$ to S_C, where $\mathsf{Pseudonyms} = \{\}$.
2. $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$ replaces **Context Key Request** with **Pseudonym Request**, where on input $(\mathsf{Pseudonym}, \xi)$ the ideal functionality samples a string η and updates Pseudonyms with η where $(U, \xi, \_, \_, \_, \mathsf{Pseudonyms}) \in S_C$. A user may request arbitrarily many such pseudonyms.
3. For **Relationship Attestation Issuance**, $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$ now identifies the issuing user by looking up the provided pseudonym η.

We sketch the proof of impossibility by considering the following specific show predicate f_{distinct}. A user provides two attestations for a statement st, and the verifier wishes to be convinced that these attestations were issued by two distinct underlying users, not merely distinct pseudonyms. Distinctness is defined with respect to the underlying credential holder associated with the pseudonym η.

Formally, the functionality computes $\mathsf{issuer}(\eta)$—the user identity mapping to η in its internal table—and evaluates the function on inputs $(\mathsf{att}_1, \mathsf{st}_1, \eta_1, \mathsf{att}_2, \mathsf{st}_2, \eta_2)$:

$$f_{\mathsf{distinct}}(\cdot, \cdot, \eta_1, \cdot, \cdot, \eta_2) = \begin{cases} 1 & \text{if } \mathsf{issuer}(\eta_1) \neq \mathsf{issuer}(\eta_2) \\ 0 & \text{o.w.} \end{cases}$$

We show that no real-world system achieving full unlinkability can securely realize $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$. We construct two scenarios that must be indistinguishable to any verifier under the definition of full unlinkability, yet where the predicate f_{distinct} evaluates differently in the ideal functionality.

Consider an adversary who controls the attestation receiver R.

- **Scenario 1 (Distinct Users):** Two distinct users U_1 and U_2 request pseudonyms η_1 and η_2 respectively from $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$. They each generate attestations for statement st and send them to R. R then attempts to show these attestations for f_{distinct}.
- **Scenario 2 (Sybil User):** A single user U_1 requests two distinct pseudonyms η'_1 and η'_2 from $\mathcal{F}_{\mathsf{PoP\text{-}full\text{-}unlinkability}}$. This user generates two attestations for statement st (one under each pseudonym) and sends them to R. R then attempts to show these attestations for f_{distinct}.

The Contradiction. In the Ideal World, the functionality has access to the internal mappings:

- In Scenario 1, $\mathsf{issuer}(\eta_1) \neq \mathsf{issuer}(\eta_2)$, so f_{distinct} outputs 1.
- In Scenario 2, $\mathsf{issuer}(\eta'_1) = \mathsf{issuer}(\eta'_2) = U_1$, so f_{distinct} outputs 0.

However, in the Real World, if the protocol satisfies *full unlinkability*, the view of the receiver R (i.e., the transcripts containing the pseudonyms and attestations) in Scenario 1 must be computationally indistinguishable from the view in Scenario 2. If R could distinguish them, R would be linking the pseudonyms in Scenario 2 to the same source, directly violating the unlinkability definition. Because the views are indistinguishable, any real-world verification algorithm run by R must output 1 with effectively the same probability in both scenarios. Thus, the real-world protocol cannot reproduce the strict distinctness check of the ideal world, leading to a failure of simulation.

Relaxation via Contexts. This impossibility result demonstrates that we cannot simultaneously have a system where users are universally unlinkable *and* where we can prove statements about the distinctness of users (sybil-resistance). To resolve this, we introduce the concept of **Contexts**. By scoping pseudonyms to a specific context (e.g., a specific application or service), we allow linkability *within* that context (enabling f_{distinct} to function correctly via context-specific keys) while maintaining unlinkability *across* different contexts.

5.4 Example f - Supporting Revocation

As an illustrative example, we describe how the predicate f can be used to support revocation, even though revocation is not natively modeled by the ideal functionality. During PHC issuance, the issuer I and user U can jointly derive a *nullifier* null that hides the identity of U. For instance, one may set $\mathsf{null} = \mathsf{H}(\mathsf{isk} || \mathsf{pk}_U)$, where pk_U is the user's ephemeral key used in its interaction with I. The nullifier is then included as an additional attribute in the issued PHC.

During VRC issuance, this nullifier is always revealed and verified as part of the VRC. A publicly available revocation list RL may contain nullifiers corresponding to revoked credentials. When verifying a VRC directly, a recipient can simply check that the nullifier is not present in RL. For show proofs, however, nullifiers are not revealed, so revocation must be enforced by incorporating a membership check against RL into the predicate f. The efficiency of this approach depends on the representation of RL and the cost of membership testing.

An important caveat is that revealing the nullifier during VRC issuance may allow the original issuer to recognize the credential, even if the protocol uses unlinkable keys. This does not violate our security definition, since any attribute explicitly revealed during issuance may enable tracking. Designing systems that natively support revocation while simultaneously maintaining strong unlinkability remains an important direction for future work.

6 Vouchable Credentials

As an intermediate primitive, we define a Vouchable Credentials, which captures a broad class of credential systems suitable for our proof-of-personhood construction. The subsequent transformation for the proof of personhood is quite generic and can be instantiated with any scheme satisfying the following definition.

Intuitively, a vouchable credential allows an issuer to issue credentials to a user, who can in turn use these credentials to generate vouches for other users. We define the scheme with respect to any non-interactive commitment scheme in the CRS model $\mathsf{Com} = (\mathsf{Setup}, \mathsf{com})$. In more detail, a user possessing a vector of attributes att interacts with an issuer identified by its public key ipk. At the end of this interaction, the user obtains a credential cred corresponding to att. The user may then reveal any subset of its attributes, $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$, specified by a set of indices idx, and use the credential cred to produce a vouch π for a statement st along with a commitment c. The resulting vouch π and c can be verified using $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx})$.

For security we require the following properties:

- *Unforgeability:* A user who receives a credential for attributes att should be unable to produce a vouch π and commitment c that verify for a set of attributes att' inconsistent with the issued att at the specified indices idx. In particular, for any idx, verification must fail if $\mathsf{att}|_{\mathsf{idx}} \neq \mathsf{att}'$, ensuring that any valid vouch corresponds to attributes for which the adversary successfully completed issuance.

- *Vouch Non-malleability:* Even when provided with oracle access to generate vouches for chosen statements and attribute subsets, an adversary should not be able to derive a new, valid vouch for a fresh statement, index, or commitment triple $(\mathsf{st}, \mathsf{idx}, c)$ not previously returned by the oracle.

- *Vouch Extractability:* We require that for any adversary producing a valid vouch, there exists a polynomial-time extractor capable of recovering the underlying attribute vector att^* and the commitment randomness r. Crucially, this property must hold even against adversarially generated issuer keys ipk, ensuring that the commitment c is binding and the proof π is well-formed regardless of whether the setup was performed honestly.

- *Vouch Simulatability:* This property ensures that the vouching process leaks no information beyond the certified attribute projections. Given the issuer's secret key isk, a stateful simulator must be able to produce commitments and proofs whose joint distribution over adaptive queries is indistinguishable from that of honestly generated vouches for a single underlying credential.

We now present the formal description. A Vouchable Credentials vCred is specified by a tuple $(\mathsf{Setup}, \mathsf{Init}, \mathsf{Issue}_{I,U}, \mathsf{Vouch}, \mathsf{Verify})$:

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{crs}, \mathsf{td})$: A randomized setup algorithm that takes as input the security parameter, and a parameter ℓ specifying the upper bound on the number of attributes. It outputs public parameters pp, crs crs and a trapdoor td. For the below algorithms, we assume implicitly that they also take in as inputs pp and crs but do not make it explicit for notational convenience.

$\mathsf{Init}(1^\lambda, 1^\ell) \to (\mathsf{isk}, \mathsf{ipk})$: A randomized initialization algorithm that takes as input the security parameter, and a parameter ℓ specifying the upper bound on the number of attributes. It outputs issuer public and private keys ipk and isk.

$\mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle \rightarrow \mathsf{cred}/\bot$**:** An interactive protocol between an issuer I and user U. The issuer has input its secret issuer key isk whereas the user has an input the issuer public key and a set of ℓ attributes att. At the end of the protocol, the user gets as output a credential cred or the issuance fails with $\bot$.

$\mathsf{Vouch}(\mathsf{st},\mathsf{att},\mathsf{idx},\mathsf{cred}) \rightarrow (c,\pi)$ A randomized algorithm which takes as input a statement st, a set of ℓ attributes att, a set of distinct indices $\mathsf{idx} \in [\ell]^\ell$ and an issued credential cred. The output of the algorithm is a commitment c and vouch π.

$\mathsf{Verify}(\mathsf{ipk},\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) \rightarrow 0/1$ A deterministic algorithm which takes as input the issuer public key ipk, statement st, a subset of attributes att′, indices idx, a vouch π and commitment c. The algorithm outputs 0 or 1 to indicate whether it rejects or accepts the vouch.

Definition 9 (Vouchable Credentials). *A vouchable credential scheme* $\mathsf{vCred} = (\mathsf{Init},\mathsf{Issue}_{I,U},\mathsf{Vouch},\mathsf{Verify})$ *must satisfy the following properties.*

Correctness: *For every* $\lambda \in \mathbb{N}$*,* $\ell \in \mathbb{N}$*,* $\mathsf{idx} \in [\ell]^\ell$*, any statement* st*, attributes* att, att′ *such that* $\mathsf{att}|_{\mathsf{idx}} = \mathsf{att}'$*:*

$$\Pr\left[\ \bot \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle\ :\ (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell)\ \right] \leq \mathsf{negl}(\lambda)$$

and

$$\Pr\left[\begin{array}{c|r} & (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ \mathsf{Verify}(\mathsf{ipk},\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) = 0\ : & \mathsf{cred} \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle \\ & (c,\pi) \leftarrow \mathsf{Vouch}(\mathsf{st},\mathsf{att},\mathsf{idx},\mathsf{cred}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Unforgeability: *For any polynomial time adversary* $\mathcal{A}$*, there exists a probabilistic polynomial time extractor* Ext *such that for all auxiliary inputs* $z \in \{0,1\}^*$

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{ipk},\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) = 1\ \wedge \\ (\mathsf{com}(\mathsf{crs},\mathsf{att}^*;r) \neq c \vee \mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}' \\ \vee \mathsf{att}^* \notin Q_{\mathsf{attr}}) \end{array} :\ \begin{array}{r} (\mathsf{pp},\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda,1^\ell) \\ (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ (\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)}(\mathsf{ipk},z) \\ (\mathsf{att}^*,r) \leftarrow \mathsf{Ext}(\mathsf{ipk},z) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)$ *is an oracle that executes the honest issuer protocol* $\mathsf{Issue}_{I,U}$ *on input attributes chosen by the adversary.* Q_{attr} *is defined as the set of all attribute vectors for which the adversary successfully completed the protocol with the issuance oracle.*

Vouch Non-Malleability: *For any stateful polynomial time adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{ipk},\mathsf{st}',\mathsf{att}|_{\mathsf{idx}'},\mathsf{idx}',\pi,c') = 1\ \wedge \\ \nexists(\mathsf{st},\mathsf{idx},c) \in Q\ s.t \\ (\mathsf{st}' = \mathsf{st} \wedge \mathsf{idx} = \mathsf{idx}' \wedge c = c') \end{array} :\ \begin{array}{r} (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ \mathsf{att},\rho_1 \leftarrow \mathcal{A}(\mathsf{ipk}) \\ \mathsf{cred} \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle \\ \pi,c',\mathsf{st}',\mathsf{idx}' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})$ *maintains a list of queries Q made by $\mathcal{A}$ and on query* (st, idx)*, updates* $Q = Q \cup (\mathsf{st},\mathsf{idx},c)$ *and responds with* Vouch(st, att, idx, cred) *where c is the commitment output by* Vouch*. Further,* $\mathsf{cred} \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle$ *indicates that an honest credential was issued for adversary specified* att.

Vouch Extractability: *For every probabilistic polynomial time adversary* $\mathcal{A}$ *there exists a probabilistic polynomial time extractor* Ext *such that for all auxiliary inputs* $z \in \{0,1\}^*$

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{ipk},\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) = 1\ \wedge \\ \mathsf{com}(\mathsf{crs},\mathsf{att}^*;r) \neq c \end{array} :\ \begin{array}{r} (\mathsf{pp},\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda,1^\ell) \\ (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ (\mathsf{ipk},\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c) \leftarrow \mathcal{A}(\mathsf{ipk},z) \\ (\mathsf{att}^*,r) \leftarrow \mathsf{Ext}(\mathsf{ipk},z) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Vouch Simulatability: *For every polynomial-time adversary* $\mathcal{A}$*, there exists a stateful polynomial-time simulator* Sim *such that for every* $1^\lambda \in \mathbb{N}$*,* $\ell \in \mathbb{N}$*, all attribute vectors* att*,*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})}(\mathsf{ipk}) = 1 \ : \ \begin{array}{r} (\mathsf{pp},\mathsf{crs},\sigma) \leftarrow \mathsf{Setup}(1^\lambda,1^\ell) \\ (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ \mathsf{cred} \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk},\mathsf{att}),(\mathsf{ipk},\mathsf{att})\rangle \end{array} \right] - \Pr\left[\mathcal{A}^{\mathsf{O}_{\mathsf{Sim}}(\mathsf{isk},\sigma,\cdot,\cdot)}(\mathsf{ipk}) = 1 \ : \ \begin{array}{r} (\mathsf{pp},\mathsf{crs},\sigma) \leftarrow \mathsf{Setup}(1^\lambda,1^\ell) \\ (\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \end{array} \right] \right| \le \mathsf{negl}(\lambda)$$

Both oracles expose the same interface to $\mathcal{A}$*: on a query* $(\mathsf{st},\mathsf{idx})$ *they return a purported vouch. Here* $\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})$ *returns the honestly generated vouch on query* st,idx*, i.e.* $\mathsf{Vouch}(\mathsf{st},\mathsf{att},\mathsf{idx},\mathsf{cred})$*, and* $\mathsf{O}_{\mathsf{Sim}}(\mathsf{isk},\sigma,\cdot,\cdot)$ *first computes the revealed subset* $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$ *and then returns the output of the simulator* $\mathsf{Sim}(\mathsf{isk},\sigma,\mathsf{st},\mathsf{idx},\mathsf{att}')$.

Here σ *denotes optional simulator-side setup information. In constructions without an explicit simulation trapdoor one may set* $\sigma = \epsilon$ *and rely only on* isk*; when a simulation trapdoor is available,* σ *may include it and* Sim *may use it directly.*

Construction of vouchable credentials. In Appendix C, we show how standard credential schemes can be upgraded to vouchable credentials using a commitment scheme and a tag-based SE-NIZK proof system. However, this generic approach is *non-black-box* in its use of cryptographic primitives and can be inefficient in practice. We therefore separately construct a vouchable credential that only makes *black-box* use of cryptographic primitives, building on structure preserving signatures and polynomial commitment schemes.

7 Proof of personhood

Our construction can support up to t different Vouchable Credentials. For convenience, we will specify our construction where each issuer is associated with a *single* Vouchable Credentials. We will not make any assumptions about issuer collusion, and thus multiple such issuers can be emulated by a single issuer.

7.1 Construction I: realizing $\mathcal{F}_{\mathsf{PoP}}$

Our first construction achieves the notion of *show unlinkability*. Here, a user utilizes the VRCs issued to them to compute a VRC show proof, which hides all of the VRCs that were used to generate the proof, revealing only the statements of the VRC. Thus, given two VRC show proofs, one cannot determine if there were any common VRCs used in the generation of the proofs, making them unlinkable.

It should be noted however that there is leakage in the form of the VRC statements. While the choice of statements that VRC is issued can leak the entirety of the VRC, our protocol guarantees no additional links can be established beyond those from the statements themselves.

7.1.1 Components

We specify the components underlying our construction

- t (potentially distinct) Vouchable Credentials (Definition 9): $\{\mathsf{vCred}^{(i)} = (\mathsf{Init}^{(i)}, \mathsf{Issue}^{(i)}_{I,U}, \mathsf{Vouch}^{(i)}, \mathsf{Verify}^{(i)})\}_{i\in[t]}$
- A SE-NIZK (Definition 5) $\mathsf{NIZK}_{\mathsf{pk}} = (\mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{pk}}$.
- A SE-NIZK $\mathsf{NIZK}_{\mathsf{ShowVRC}} = (\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{ShowVRC}}$.

Remark 4. *While the NIZKs are defined to be tag-based (Definition 5), in this section we omit the tag for readability.*

7.1.2 Proof of Knowledge Languages

We describe here the various languages that will be used in our protocol.

Proof of Knowledge of Secret Key. The user will generate a zkPoK for the language $\mathcal{L}_{\mathsf{pk}}$ to establish ownership of the secret-key corresponding to the public key, i.e.

$$\mathcal{L}_{\mathsf{pk}} = \{\mathsf{pk} \mid \exists \mathsf{sk} \text{ s.t. } (\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}\}.$$

We assume here that it is efficient to test whether $(\mathsf{pk}, \mathsf{sk})$ is a valid public-key secret-key pair.

Proof of knowledge of VRC. A user who has collected many VRCs generates a zkPoK for the language $\mathcal{L}_{\mathsf{ShowVRC}}$, establishing that for claimed statements $\mathsf{st}_1, \ldots, \mathsf{st}_N$ and a predicate f: (i) the user holds valid VRCs issued with attributes $\mathsf{att}_1, \ldots, \mathsf{att}_N$ respectively; (ii) $f(\mathsf{att}_1, \mathsf{st}_1, \mathsf{pk}_1 \ldots, \mathsf{att}_N, \mathsf{st}_N, \mathsf{pk}_N) = 1$.

We formally specify these constraints below, and note that the language is also parameterized by the issuer keys $\{\mathsf{ipk}^{(i)}\}_{i\in[t]}$, but for ease of exposition we omit this in the language.

$$\begin{aligned}\mathcal{L}_{\mathsf{ShowVRC}} = \Big\{ &(\mathsf{pk}, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N)) \mid \exists \{\mathsf{att}_j, \mathsf{pk}_j, \mathsf{idx}_j, \pi_j\}_{j\in[N]} \text{ s.t.} \\ &\forall j \in [N], \mathsf{VerifyVRC}(\mathsf{ipk}^{(i_j)}, \mathsf{pk}, \mathsf{st}_j, \mathsf{pk}_j, \mathsf{att}_j, \mathsf{idx}_j, \pi_j) = 1 \wedge \text{ //Statements have a VRC} \\ &f(\mathsf{att}_1, \mathsf{st}_1, \mathsf{pk}_1 \ldots, \mathsf{att}_N, \mathsf{st}_N, \mathsf{pk}_N) = 1 \text{ //Statements and attributes satisfy predicate } f \Big\}\end{aligned}$$

7.1.3 Construction

The construction is specified in Fig. 2.

Proofs of Personhood

Parameters: Security paramater 1^λ, number of credentials supported 1^t, number of attributes supported within each credential 1^ℓ. A set of t issuers, $\{I^{(i)}\}_{i\in[t]}$.

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^t)$:

```
  1. For each $i \in [t]$, $(\mathsf{ipk}^{(i)}, \mathsf{isk}^{(i)}) \leftarrow \mathsf{Init}^{(i)}(1^\lambda, 1^{\ell+1})$ //Additional for the public key
  2. $\mathsf{crs}_{\mathsf{pk}}, \_ \leftarrow \mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Setup}(1^\lambda)$, $\mathsf{crs}_{\mathsf{ShowVRC}}, \_ \leftarrow \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Setup}(1^\lambda)$
  3. Set $\mathsf{pp} = (\mathsf{crs}_{\mathsf{pk}}, \mathsf{crs}_{\mathsf{ShowVRC}}\{\mathsf{ipk}^{(i)}\}_{i\in[t]})$.
  4. Output $(\mathsf{pp}, \{\mathsf{isk}^{(i)}\}_{i\in[t]})$.
```

$\mathsf{IssuePHC}(\langle I^{(i)}(\mathsf{pp}, \mathsf{isk}^{(i)}), U(\mathsf{pp}, \mathsf{ipk}^{(i)}, \mathsf{att}_U, \mathsf{pk}_U, \mathsf{sk}_U)\rangle)$:

```
  $U \to I^{(i)}$: //User proves ownership of public key
    1. $U$ computes $\pi_{\mathsf{NIZK,pk}} \leftarrow \mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{pk}}, \mathsf{pk}_U, \mathsf{sk}_U)$
    2. Send $\pi_{\mathsf{NIZK,pk}}$ along with $\mathsf{pk}_U$ and $\mathsf{att}_U$ to $I^{(i)}$.
  $U \leftrightarrow I^{(i)}$ //User and Issuer run underlying issuance protocol with additional fixed attributes $\mathsf{pk}_U$
    1. $I^{(i)}$ outputs $\perp$ if $\mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{pk}}, \mathsf{pk}_U, \pi_{\mathsf{NIZK,pk}}) = 0$.
    2. Set $\mathsf{att} = \mathsf{pk}_U || \mathsf{att}_U$
    3. $U$ and $I^{(i)}$ run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\mathsf{isk}^{(i)}, \mathsf{att}), (\mathsf{ipk}^{(i)}, \mathsf{att})\rangle$.
    4. $U$ obtains $\mathsf{cred}^{(i)}_U$ at the end of the protocol.
```

$\mathsf{IssueVRC}(\mathsf{pp}, \mathsf{cred}^{(i)}_U, \mathsf{att}_U, \mathsf{idx}, \mathsf{pk}_{U'}, \mathsf{st})$:

1. Set $\mathsf{idx}' = \mathsf{idx} \cup \{1\}$, $\mathsf{st}' = \mathsf{st}||\mathsf{pk}_{U'}$, $\mathsf{att} = \mathsf{pk}_U||\mathsf{att}_U$ //Always reveal public key during a VRC issuance and tie statement proven to $\mathsf{pk}_{U'}$
2. Compute $\pi_U^{(i)}, c_U^{(i)} \leftarrow \mathsf{Vouch}^{(i)}(\mathsf{st}', \mathsf{att}, \mathsf{idx}', \mathsf{cred}_U^{(i)})$
3. Output $\pi_U^{(i)}$, $c_U^{(i)}$, $\mathsf{ipk}^{(i)}$, st, idx and $\mathsf{att}|_{\mathsf{idx}}$ //Include the key of the issuer to verify VRC

$\underline{\mathsf{VerifyVRC}(\mathsf{pp}, \mathsf{ipk}^{(i)}, \mathsf{pk}_{U'}, \mathsf{st}, \mathsf{pk}_U, \mathsf{att}, \mathsf{idx}, \pi_U^{(i)}, c_U^{(i)})}$:

1. Set $\mathsf{idx}' = \mathsf{idx} \cup \{1\}$, $\mathsf{st}' = \mathsf{st}||\mathsf{pk}_{U'}$, $\mathsf{att}' = \mathsf{pk}_U||\mathsf{att}$
2. Output $\mathsf{Verify}^{(i)}(\mathsf{ipk}^{(i)}, \mathsf{st}', \mathsf{att}', \mathsf{idx}', \pi_U^{(i)}, c_U^{(i)})$ //Verify underlying vouch

$\underline{\mathsf{ShowVRC}(\mathsf{pp}, f, \mathsf{pk}_U, \{\mathsf{ipk}^{(i_j)}, \mathsf{st}_j, \mathsf{pk}_j, \mathsf{att}_j, \mathsf{idx}_j, \pi_j\}_{j\in[N]})}$:

1. Set $\mathsf{x}_{\mathsf{ShowVRC}} = (\mathsf{pk}_U, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N))$ //Generate proof of $\mathcal{L}_{\mathsf{ShowVRC}}$
2. Set $\mathsf{w}_{\mathsf{ShowVRC}} = \{\mathsf{att}_j, \mathsf{pk}_j, \mathsf{idx}_j, \pi_j\}_{j\in[N]}$
3. Compute $\pi_{\mathsf{ShowVRC}} \leftarrow \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}, \mathsf{w}_{\mathsf{ShowVRC}})$.
4. Output $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$.

$\underline{\mathsf{VerifyShow}(\mathsf{pp}, \mathsf{pk}_U, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N), \pi_{\mathsf{ShowVRC}})}$:

1. Set $\mathsf{x}_{\mathsf{ShowVRC}} = (\mathsf{pk}_U, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N))$
2. Output $\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}, \pi_{\mathsf{ShowVRC}})$.

Figure 2: Proof of Personhood realizing $\mathcal{F}_{\mathsf{PoP}}$.

Theorem 6 (Realization of $\mathcal{F}_{\mathsf{PoP}}$ with PAV Oracles). *Assume the security of the underlying vouchable-credential scheme (Definition 9) and the non-interactive zero-knowledge proof system (Definition 5). Then the proof-of-personhood system*

$$(\mathsf{Setup}, \mathsf{IssuePHC}, \mathsf{IssueVRC}, \mathsf{VerifyVRC}, \mathsf{ShowVRC}, \mathsf{VerifyShow})$$

above securely realizes the ideal functionality $\mathcal{F}_{\mathsf{PoP}}$ in the real/ideal world model with respect to the PAV oracle ensemble defined in Section 4.

7.1.4 Security Proof

Simulator. We now describe the simulation strategy for our protocol. In particular, for every adversary $\mathcal{A}$ we construct a simulator $\mathcal{S}$ that interacts with $\mathcal{A}$ and $\mathcal{F}_{\mathsf{PoP}}$. $\mathcal{E}$ provides inputs to the honest parties, and the messages of $\mathcal{E}$. In this work it will suffice to think of $\mathcal{A}$ as simply relaying the messages provided by $\mathcal{E}$, which may not follow the protocol specification, i.e. might be maliciously generated.

Let the set of corrupted parties be C. We now describe the simulator $\mathcal{S}$, separating credential issuance, VRC issuance and showing. $\mathcal{S}$ maintains a list of credentials CredList and list of attestations AttList, both initialized to be empty. These lists keep track of the credentials and attestations issued by the ideal functionality $\mathcal{F}_{\mathsf{PoP}}$. However, since adversarial users and issuers may arbitrarily issue each other credentials and attestations, circumventing $\mathcal{F}_{\mathsf{PoP}}$, when these are used to issue attestations or showing to honest users, i.e. when $\mathcal{S}$ is made aware of them, $\mathcal{S}$ must "register" these credentials and attestations with $\mathcal{F}_{\mathsf{PoP}}$ by initiating a new credential (or attestation) issuance, playing the role of both issuer and receiver, without forwarding any messages to $\mathcal{A}$.

For a candidate VRC tuple $(\mathsf{pk}_{\mathsf{recv}}, \mathsf{pk}_{\mathsf{iss}}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$, we say there is a *matching attestation record* in AttList if there exists $(\mathsf{id}, \mathsf{pk}_{\mathsf{recv}}, \mathsf{pk}_{\mathsf{iss}}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)}) \in \mathsf{AttList}$.

Looking ahead, we will delineate special events E with a symbol $\perp_E$, which will be useful when reducing to the security of the underlying primitives.

Setup. $\mathcal{S}$ samples the crs for the NIZK, along with the trapdoor. $\mathcal{S}$ also samples keys on behalf of the honest issuers and users.

1. $\mathsf{crs_{pk}}, \mathsf{td_{pk}} \leftarrow \mathsf{NIZK_{pk}.Setup}(1^\lambda)$.
2. $\mathsf{crs_{ShowVRC}}, \mathsf{td_{ShowVRC}} \leftarrow \mathsf{NIZK_{ShowVRC}.Setup}(1^\lambda)$.
3. For each $I^{(i)} \notin C$, $(\mathsf{ipk}^{(i)}, \mathsf{isk}^{(i)}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
4. For each $U \notin C$, $(\mathsf{pk}_U, \mathsf{sk}_U) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
5. Initialize $\mathsf{CredList} \leftarrow \emptyset$ and $\mathsf{AttList} \leftarrow \emptyset$.

Credential Issuance. For the issuance of a credential, $\mathcal{S}$ only needs to simulate the interaction between an issuer and user if exactly one of them are corrupted.

- If the credential recipient user $U \in C$:
  1. Receive from $\mathcal{A}$ (on behalf of U) $\pi_{\mathsf{NIZK,pk}}$, pk_U and att_U intended for honest issuer $I^{(i)}$.
  2. $\mathcal{S}$ outputs $\perp$ if $\mathsf{NIZK_{pk}.Verify}(\mathsf{crs_{pk}}, \mathsf{pk}_U, \pi_{\mathsf{NIZK,pk}}) = 0$.
  3. $\mathcal{S}$ uses $\mathsf{NIZK_{pk}.Ext}$ to extracts from $\pi_{\mathsf{NIZK,pk}}$ the secret key sk and checks if $(\mathsf{pk}_U, \mathsf{sk}) \in \mathsf{KeyGen}$. If this fails, output $\perp_{\mathsf{NIZK}}$ and abort.
  4. Send $(\mathsf{credIssuance}, I^{(i)}, \mathsf{att}_U)$ to $\mathcal{F}_{\mathsf{PoP}}$.
  5. Forward $\mathcal{U}$ from U to $O_\lambda^{(i)}$ unchanged.
  6. If $\mathcal{F}_{\mathsf{PoP}}$ sends $\mathsf{credApproved}$, run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\mathsf{isk}^{(i)}, \mathsf{att}_U), (\cdot,\cdot)\rangle$ with U, behaving honestly on behalf of $I^{(i)}$.
  7. Add $(U, I^{(i)}, \mathsf{pk}_U, \mathsf{att}_U)$ to $\mathsf{CredList}$.
- If the credential issuer $I^{(i)} \in C$:
  1. $\mathcal{S}$ receives $(\mathsf{credIssuance}, U, \mathsf{att})$ from $\mathcal{F}_{\mathsf{PoP}}$.
  2. $\mathcal{S}$ simulates $\pi_{\mathsf{NIZK,pk}} \leftarrow \mathsf{NIZK_{pk}.SimProve}(\mathsf{crs_{pk}}, \mathsf{td_{pk}}, \mathsf{pk}_U)$. Sends $\pi_{\mathsf{NIZK,pk}}$, pk_U and att to $\mathcal{A}$.
  3. Forward att' from $\mathcal{A}$ to $O_\lambda^{(i)}$ and return the response.
  4. Run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\cdot,\cdot), (\mathsf{ipk}^{(i)}, \mathsf{att})\rangle$ with $I^{(i)}$ honestly on behalf of U.

VRC Issuance. $\mathcal{S}$ simulates interaction of the VRC issuance between an honest and corrupted user.

- If the VRC recipient user $U \in C$:
  1. Receive from $\mathcal{A}$ (on behalf of U) st for VRC issuance from user V.
  2. $\mathcal{S}$ sends $(\mathsf{relAttestIssuance}, \mathsf{st}, V)$ to $\mathcal{F}_{\mathsf{PoP}}$.
  3. On receiving $(\mathsf{relAttestApproved}, \mathsf{att}', \mathsf{idx}, I^{(i)}, \mathsf{id})$ from $\mathcal{F}_{\mathsf{PoP}}$, simulate the vouch π. If $I^{(i)} \in C$, then $\mathcal{S}$ has access to both the corresponding att and cred, and it can generate the vouch honestly. However if $I^{(i)} \notin C$, $\mathcal{S}$ uses the vouch simulator Sim, and outputting $\mathsf{Sim}(\mathsf{isk}^{(i)}, \mathsf{st}, \mathsf{idx}, \mathsf{att}')$.
  4. Add $(\mathsf{id}, \mathsf{pk}_U, \mathsf{pk}_V, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to $\mathsf{AttList}$.
  5. Send π, $\mathsf{ipk}^{(i)}$, st, $\mathsf{idx} \cup \{1\}$ and $\mathsf{pk}_V||\mathsf{att}'$ to $\mathcal{A}$.
- If the VRC issuer user $U \in C$:
  1. $\mathcal{S}$ receives $(\mathsf{relAttestIssuance}, V, \mathsf{st})$ from $\mathcal{F}_{\mathsf{PoP}}$.
  2. $\mathcal{S}$ sends to V (via $\mathcal{A}$) the statement st.
  3. On receiving π, c, $\mathsf{ipk}^{(i)}$, st, idx, and $\mathsf{pk}_U||\mathsf{att}'$ from $\mathcal{A}$, set $\mathsf{idx}' = \mathsf{idx} \cup \{1\}$ and $\mathsf{st}' = \mathsf{st}$, then check $\mathsf{Verify}^{(i)}(\mathsf{ipk}^{(i)}, \mathsf{st}', \mathsf{pk}_U||\mathsf{att}', \mathsf{idx}', \pi, c) = 1$.
  4. Runs Ext of the vouchable credential to obtain att^*, r from the vouch π.
  5. If $\mathsf{com}(\mathsf{crs}, \mathsf{att}^*; r) \neq c$ or $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$, abort with $\perp_{\mathsf{VouchExtract}}$.
  6. If $I^{(i)} \in C$, $\mathcal{S}$ need to send the credential to $\mathcal{F}_{\mathsf{PoP}}$ simulating the interaction on behalf of both U and $I^{(i)}$.
     (a) Sends $(\mathsf{credIssuance}, I^{(i)}, \mathsf{att})$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of U.

(b) Receive $(\mathsf{credIssuance}, U, \mathsf{att})$ from $\mathcal{F}_{\mathsf{PoP}}$ on behalf of $I^{(i)}$.

(c) Sends $(\mathsf{credApproved}, U)$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of $I^{(i)}$. //Since the credential was already issued, there is no need to query $O_{\lambda}^{(i)}$

(d) Add $(U, I^{(i)}, \mathsf{pk}_U, \mathsf{att})$ to CredList.

7. If $I^{(i)} \notin C$, $\mathcal{S}$ checks if $(U, I^{(i)}, \mathsf{pk}_U, \mathsf{att}) \in \mathsf{CredList}$. If not, abort with $\perp_{\mathsf{PHCforgery}}$.
8. Else send $(\mathsf{approveRelAttest}, V, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to $\mathcal{F}_{\mathsf{PoP}}$.
9. On receiving $(\mathsf{relAttestApproved}, \mathsf{id})$ from $\mathcal{F}_{\mathsf{PoP}}$, add $(\mathsf{id}, \mathsf{pk}_V, \mathsf{pk}_U, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to AttList.

Showing. $\mathcal{S}$ simulates interaction of the showing between an honest and corrupted user.

- If the show recipient user $U \in C$:
  1. $\mathcal{S}$ receives $(\mathsf{showCred}, V, f, (\mathsf{st}_1, I^{(i_1)}, \ldots, \mathsf{st}_N, I^{(i_N)}))$ from $\mathcal{F}_{\mathsf{PoP}}$.
  2. $\mathcal{S}$ sets $\mathsf{x}_{\mathsf{ShowVRC}} = (\mathsf{pk}_U, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N))$ and simulates π_{ShowVRC}.
  3. Send $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$ to $\mathcal{A}$.
- If the show issuer user $U \in C$:
  1. $\mathcal{S}$ receives $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$ from $\mathcal{A}$.
  2. $\mathcal{S}$ aborts if π_{ShowVRC} does not verify.
  3. $\mathcal{S}$ uses $\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Ext}$ to extract $\mathsf{w}_{\mathsf{ShowVRC}} = \{\mathsf{att}_j, \mathsf{pk}_j, \mathsf{idx}_j, \pi_j\}_{j \in [N]}$ and output $\perp_{\mathsf{NIZK}}$ if $\mathsf{w}_{\mathsf{ShowVRC}}$ is not a valid witness for the statement $\mathsf{x}_{\mathsf{ShowVRC}}$ in $\mathcal{L}_{\mathsf{ShowVRC}}$.
  4. For each $j \in [N]$, let $(\mathsf{att}_j, \mathsf{pk}_j, \mathsf{idx}_j, \pi_j)$ be the extracted tuple corresponding to issuer $I^{(i_j)}$.

     (a) If $I^{(i_j)} \notin C$ and the user corresponding to pk_j is honest, $\mathcal{S}$ checks that there exists a matching attestation record in AttList for $(\mathsf{pk}_U, \mathsf{pk}_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$. If no such record exists, abort with $\perp_{\mathsf{VRCforgery}}$.

     (b) If $I^{(i_j)} \notin C$ but the user corresponding to pk_j is corrupted, $\mathcal{S}$ may not have seen the VRC before. In this case, $\mathcal{S}$ checks whether the extracted underlying credential corresponds to a PHC that was previously issued and recorded in CredList. If not, abort with $\perp_{\mathsf{PHCforgery}}$. Otherwise, $\mathcal{S}$ invokes $\mathcal{F}_{\mathsf{PoP}}$ to issue the corresponding relationship attestation, receives identifier id_j, and adds $(\mathsf{id}_j, \mathsf{pk}_U, \mathsf{pk}_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$ to AttList.

     (c) If $I^{(i_j)} \in C$ but the user corresponding to pk_j is honest, $\mathcal{S}$ checks that there exists a matching attestation record in AttList for $(\mathsf{pk}_U, \mathsf{pk}_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$. If no such record exists, abort with $\perp_{\mathsf{VRCforgery}}$.

     (d) If both $I^{(i_j)} \in C$ and the user corresponding to pk_j is corrupted, $\mathcal{S}$ simulates the corresponding relationship-attestation issuance in $\mathcal{F}_{\mathsf{PoP}}$ on behalf of the corrupted parties, receives identifier id_j, and adds $(\mathsf{id}_j, \mathsf{pk}_U, \mathsf{pk}_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$ to AttList.
  5. Let $\mathsf{id}_1, \ldots, \mathsf{id}_N$ be the attestation identifiers returned by $\mathcal{F}_{\mathsf{PoP}}$ (or recorded earlier) for each of the above checks. $\mathcal{S}$ sends $(\mathsf{showCred}, V, f, (\mathsf{id}_1, \ldots, \mathsf{id}_N))$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of the corrupted user.

Hybrids. We describe below a sequence of *hybrid* experiments, where we go from the execution of the real world, to an execution of the protocol emulated by the simulator above. We will make a small change in each of the hybrids, and argue that the change made was (computationally) indistinguishable to the environment. In particular, we denote by Hyb_i the output of the environment in that particular hybrid. It then suffices to show for each adjacent hybrid, $\mathsf{Hyb}_i \approx_c \mathsf{Hyb}_{i+1}$, i.e. the hybrids are computationally indistinguishable.

Hyb_0: *Real world.* This is the real execution of the protocol with honest parties running the specified algorithms and corrupted parties controlled by $\mathcal{A}$. $\mathcal{S}$ can emulate this by behaving honestly on behalf of all honest parties using their inputs.

Hyb_1: *Simulated NIZKs for honest parties.* Modify the execution so that all NIZK proofs generated by honest parties are produced using the NIZK simulator and the simulation trapdoor rather than by running the honest prover.

Claim 1. $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$

Proof. Indistinguishability follows from the (multi-theorem) computational zero-knowledge property of the NIZK. We make the changes for the hybrid in two steps, in the first step, represented by $\mathsf{Hyb}_{1,\mathsf{pk}}$, replacing all the $\mathsf{NIZK}_{\mathsf{pk}}$ with the simulated proofs, and then correspondingly for $\mathsf{NIZK}_{\mathsf{ShowVRC}}$. We show that each step is indistinguishable.

In particular, we show that if there exists an adversary $\mathcal{A}$ that is able to cause $\mathsf{Hyb}_{1,\mathsf{pk}}$ and Hyb_0 to be distinguishable, then we can construct an adversary $\mathcal{D}_{\mathsf{ZK}}$ that violates the zero-knowledge property of $\mathsf{NIZK}_{\mathsf{pk}}$. The indistinguishability of the next step (replacing the proof for $\mathsf{NIZK}_{\mathsf{ShowVRC}}$) follows in an analogous manner. And thus by a simple triangle inequality we have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0$. We now show $\mathsf{Hyb}_{1,\mathsf{pk}} \approx_c \mathsf{Hyb}_0$

$\underline{\mathcal{D}_{\mathsf{ZK}}^{\mathsf{O}}(\mathsf{crs})}$

1. Emulate the execution of Hyb_0 with $\mathcal{A}$ using honest inputs, except for generation of proof for $\mathsf{NIZK}_{\mathsf{pk}}$.
2. Whenever an honest party must generate a NIZK for $\pi_{\mathsf{NIZK},\mathsf{pk}}$, it queries the oracle O on $(\mathsf{pk}, \mathsf{sk})$ and uses the returned proof as $\pi_{\mathsf{NIZK},\mathsf{pk}}$.
3. Output the final bit output by the environment $\mathcal{E}$ in this emulated execution.

If O is $\mathsf{O}_{\mathsf{real}}$, then the execution corresponds exactly to Hyb_0, however if O is $\mathsf{O}_{\mathsf{Sim}}$, then it corresponds to the execution of $\mathsf{Hyb}_{1,\mathsf{pk}}$. Thus if $\mathcal{E}$ can distinguish the two hybrids (in the presence of $\mathcal{A}$), then $\mathcal{D}_{\mathsf{ZK}}$ breaks the zero-knowledge property of the NIZK.

$\square$

Hyb_2: *Extraction from NIZKs.* Modify the execution so that whenever a corrupted party sends a NIZK proof, $\mathcal{S}$ runs the extractor to obtain the corresponding witness. If the proof verifies but the extracted witness is invalid or inconsistent with the statement, $\mathcal{S}$ aborts with $\perp_{\mathsf{NIZK}}$. All other behavior remains unchanged.

Claim 2. $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$

Proof. Indistinguishability follows from the (multi-theorem) weak simulation-extractability property of the NIZK proof systems.

The only difference between the two hybrids is that in Hyb_1, when a corrupted party sends an accepting NIZK proof, the execution continues, whereas in Hyb_2, $\mathcal{S}$ additionally runs the extractor and aborts with $\perp_{\mathsf{NIZK}}$ if the extracted witness is invalid/inconsistent.

Thus, if the hybrids are distinguishable with non-negligible advantage, then with non-negligible probability there is an execution in which a corrupted party outputs an accepting proof that causes $\perp_{\mathsf{NIZK}}$. We use this to build an adversary $\mathcal{B}_{\mathsf{SE}}$ against weak simulation-extractability.

$\underline{\mathcal{B}_{\mathsf{SE}}^{\mathsf{SimProve}(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs})}$

1. Emulate Hyb_1 with $\mathcal{A}$ using honest inputs.
2. Whenever an honest party needs a simulated NIZK, query the simulation oracle $\mathsf{SimProve}$ and return the resulting proof.
3. Maintain the set Q of statements x queried to the simulation oracle.
4. If $\mathcal{A}$ outputs an accepting proof (x, π) such that either extraction fails or the extracted witness is invalid/inconsistent and $x \notin Q$, output (x, π).
5. Otherwise output $\perp$.

If the hybrids were distinguishable with non-negligible advantage, then with non-negligible probability the abort event $\perp_{\mathsf{NIZK}}$ occurs, which exactly yields an accepting fresh proof of the above type. This gives a break of weak simulation-extractability. The argument applies to both $\mathsf{NIZK}_{\mathsf{pk}}$ and $\mathsf{NIZK}_{\mathsf{ShowVRC}}$. Hence $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

$\square$

Hyb_3: *Recording issued PHCs and VRCs.* For every PHC issued by an honest issuer to a corrupted user, record the tuple

$$(U, I^{(i)}, \mathsf{pk}_U, \mathsf{att})$$

in a list CredList. Further, for every relationship attestation/VRC issuance observed by $\mathcal{S}$, record the exact tuple

$$(\mathsf{id}, \mathsf{pk}_{\mathsf{recv}}, \mathsf{pk}_{\mathsf{iss}}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$$

in a list AttList, where id is the identifier returned by $\mathcal{F}_{\mathsf{PoP}}$. This bookkeeping does not change any messages sent to $\mathcal{A}$ and only records information locally.

Claim 3. $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$

Proof. The two hybrids are identically distributed: the only change is additional local state updates to CredList and AttList, with no change to any message delivered to $\mathcal{A}$, honest parties, or the environment. □

Hyb_4: *Abort on vouch-extraction inconsistency.* Whenever $\mathcal{S}$ runs the vouch extractor on an accepted tuple $(\mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$, let extraction return (att^*, r) and abort with $\perp_{\mathsf{VouchExtract}}$ if either

$$\mathsf{com}(\mathsf{crs}, \mathsf{att}^*; r) \neq c \qquad \text{or} \qquad \mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'.$$

Claim 4. $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$

Proof. We reduce to vouch extractability. The only difference between the two hybrids is that Hyb_4 aborts when the event $\perp_{\mathsf{VouchExtract}}$ occurs. Thus it suffices to show this event happens with negligible probability.

Assume toward contradiction that $\Pr\left[\ \perp_{\mathsf{VouchExtract}}\ \right]$ is non-negligible. We construct an adversary $\mathcal{B}_{\mathsf{ext}}$ for the extractability game.

$\underline{\mathcal{B}_{\mathsf{ext}}(\mathsf{ipk})}$

1. Emulate Hyb_3 with $\mathcal{A}$ (including all prior hybrid checks).
2. Continue until the first execution point where $\perp_{\mathsf{VouchExtract}}$ would be triggered, i.e., for an accepted tuple $(\mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$ whose extracted witness (att^*, r) satisfies $\mathsf{com}(\mathsf{crs}, \mathsf{att}^*; r) \neq c$ or $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$.
3. Output the corresponding tuple $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$.

By construction, this output is accepted yet violates the extractability consistency condition, contradicting vouch extractability. Hence $\Pr\left[\ \perp_{\mathsf{VouchExtract}}\ \right]$ is negligible, and therefore $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$. □

Hyb_5: *Abort on forged PHCs.* Whenever a corrupted party presents a VRC or show witness claiming attributes derived from a PHC issued by an honest issuer $I^{(i)}$, extract the underlying attribute vector using the vouch extractor. If the extracted attribute vector does not appear in CredList, abort with $\perp_{\mathsf{PHCforgery}}$.

Claim 5. $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$

Proof. The only difference between the two hybrids is the following additional check: if a corrupted party presents a VRC/show witness that claims support from a PHC under an honest issuer, then $\mathcal{S}$ verifies that the corresponding extracted credential information is consistent with a previously recorded honest-issuer PHC in CredList; otherwise $\mathcal{S}$ aborts with $\perp_{\mathsf{PHCforgery}}$.

From Claim 4, we may assume extracted vouch tuples already satisfy commitment/projection consistency.

From the simulator description, this abort can arise in multiple places (in particular, during corrupted-user VRC issuance handling and during corrupted-user show handling). Let $\mathsf{E}_{\mathsf{PHC}}$ denote the union of all such PHC-forgery abort events across those branches.

If $\Pr[\ \mathsf{E}_{\mathsf{PHC}}\ ]$ were non-negligible, then by a union bound there exists at least one branch where the corresponding PHC-forgery event occurs with non-negligible probability. We build $\mathcal{B}_{\mathsf{uf}}$ against unforgeability of vCred for that branch. In this reduction, only one issuer (the issuer in the selected branch) is exposed externally via the unforgeability challenger; all other issuer instances are generated and handled internally by $\mathcal{B}_{\mathsf{uf}}$ in the emulation.

$\underline{\mathcal{B}_{\mathsf{uf}}^{\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)}(\mathsf{ipk})}$

1. Emulate the execution with $\mathcal{A}$, answering PHC issuance queries for the externally exposed issuer using the oracle $\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)$, and simulating all other issuer instances internally. Record all issued attribute vectors in Q_{attr}.
2. Continue the emulation until the first PHC-forgery abort event in the selected branch occurs.
3. Extract the corresponding verifying tuple $(\mathsf{st},\mathsf{att}',\mathsf{idx},\pi,c)$ and output it as the unforgeability forgery.

By construction, the tuple verifies and is accepted by the protocol checks, yet it does not correspond to any honestly issued attribute vector recorded in Q_{attr} (equivalently, extraction yields (att^*,r) with $\mathsf{att}^* \notin Q_{\mathsf{attr}}$, or $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$, or $\mathsf{com}(\mathsf{crs},\mathsf{att}^*;r) \neq c$), contradicting unforgeability of vCred.
Hence $\Pr[\ \mathsf{E}_{\mathsf{PHC}}\ ]$ is negligible, and therefore $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$. □

Hyb_6: *Abort on unseen honest VRCs.* Whenever a show proof includes a VRC attributed to an honest user under an honest issuer, check that the corresponding attestation record

$$(\mathsf{id},\mathsf{pk}_{\mathsf{recv}},\mathsf{pk}_{\mathsf{iss}},\mathsf{st},\mathsf{att}',\mathsf{idx},I^{(i)})$$

appears in AttList. If no such issuance record exists, abort with $\bot_{\mathsf{VRCforgery}}$.

Claim 6. $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$

Proof. The only difference between the two hybrids is the additional abort on missing honest-VRC issuance records in AttList.

Bridge to non-malleability freshness. In the exposed branch of the reduction, every honest oracle-generated vouch corresponding to $(\mathsf{st},\mathsf{idx},c)$ is recorded consistently in both (i) the oracle-query set Q and (ii) AttList via the tuple $(\mathsf{id},\mathsf{pk}_{\mathsf{recv}},\mathsf{pk}_{\mathsf{iss}},\mathsf{st},\mathsf{att}',\mathsf{idx},I^{(i)})$. Therefore, if a candidate honest VRC in that branch has no matching attestation record in AttList, then its corresponding triple $(\mathsf{st},\mathsf{idx},c)$ is fresh with respect to Q.

From the simulator description, $\bot_{\mathsf{VRCforgery}}$ can arise in multiple places during show handling (corresponding to different honest-issuer/honest-attester combinations). Let $\mathsf{E}_{\mathsf{VRC}}$ denote the union of all such unseen-honest-VRC abort events.

If $\Pr[\ \mathsf{E}_{\mathsf{VRC}}\ ]$ were non-negligible, then by a union bound there exists at least one branch where this event occurs with non-negligible probability. We build $\mathcal{B}_{\mathsf{nm}}$ against vouch non-malleability for that branch.

$\underline{\mathcal{B}_{\mathsf{nm}}^{\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})}(\mathsf{ipk})}$

1. Emulate the execution with $\mathcal{A}$, using the vouch oracle for the externally exposed honest attester/issuer branch and simulating all other branches internally.
2. Maintain the set Q of queried triples $(\mathsf{st},\mathsf{idx},c)$ corresponding to oracle-generated honest VRCs in the selected branch.
3. Continue until the first unseen-honest-VRC abort event in the selected branch occurs; extract the corresponding verifying tuple $(\mathsf{st}',\mathsf{att}',\mathsf{idx}',\pi',c')$ and output it.

By construction, the output verifies but its corresponding triple is fresh relative to Q, yielding a break of vouch non-malleability. Hence $\Pr[\ \mathsf{E}_{\mathsf{VRC}}\ ]$ is negligible, and therefore $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$. □

Hyb_7: *Replace honest vouches with simulated vouches.* Modify the execution so that whenever a vouch is generated on behalf of an honest PHC issuer $I^{(i)}$, instead of computing

$$\mathsf{Vouch}(\mathsf{st}, \mathsf{att}, \mathsf{idx}, \mathsf{cred}),$$

the challenger generates the vouch using the vouch simulator

$$\mathsf{Sim}(\mathsf{isk}^{(i)}, \mathsf{st}, \mathsf{idx}, \mathsf{att}'),$$

where $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$ is the revealed subset. For corrupted issuers, vouches continue to be generated honestly.

Claim 7. $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$

Proof. The only difference between the two hybrids is that, for honest issuers, vouches are generated using $\mathsf{Sim}(\mathsf{isk}^{(i)}, \mathsf{st}, \mathsf{idx}, \mathsf{att}')$ instead of the honest algorithm $\mathsf{Vouch}(\mathsf{st}, \mathsf{att}, \mathsf{idx}, \mathsf{cred})$.

If these hybrids were distinguishable with non-negligible advantage, then there exists at least one honest-issuer branch where replacing real vouches by simulated vouches changes the view with non-negligible probability. We build $\mathcal{B}_{\mathsf{sim}}$ against vouch simulatability for that branch.

$\underline{\mathcal{B}_{\mathsf{sim}}^{\mathsf{O}(\cdot,\mathsf{att},\cdot,\mathsf{cred})}(\mathsf{ipk})}$

1. Emulate the execution with $\mathcal{A}$, using oracle O for vouch generation in the externally exposed honest-issuer branch and simulating all other branches internally.
2. On each vouch query $(\mathsf{st}, \mathsf{idx})$ in that branch, forward the query to O and return the response.
3. Output whatever $\mathcal{A}$ outputs at the end of the emulation.

If $\mathsf{O} = \mathsf{O}_{\mathsf{Vouch}}$, the emulation is distributed as Hyb_6; if $\mathsf{O} = \mathsf{O}_{\mathsf{Sim}}$, it is distributed as Hyb_7. Hence a non-negligible distinguisher would violate vouch simulatability. Therefore $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$. □

Hyb_8: *Ideal world.* This hybrid is identical to the behavior of the simulator $\mathcal{S}$ interacting with the ideal functionality. All NIZKs are simulated, all vouches for honest issuers are simulated, and all checks and bookkeeping are enforced as in the simulator description.

Claim 8. $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$

Proof. The two hybrids are identically distributed: by this point all changes required by the simulator description have already been applied (simulated NIZKs, extraction checks, PHC/VRC bookkeeping, PHC/VRC forgery abort checks, and simulated honest-issuer vouches). Hence this hybrid is exactly the ideal-world simulator execution, and no additional distributional change occurs. □

7.2 Construction II: realizing $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$

In the prior construction, each user was in possession of a long term public key pk, which was used to obtain personhood credentials from the many issuers in the system. However, since the issuers learn the long term key from the requesting user, colluding issuers can link users across different credentials.

In this section, we want to make the issued credentials *unlinkable* by having each user sample a fresh key for each interaction with the issuer. But to allow a user assert claims about credentials and VRCs, it must also prove that the these keys belong to them. We do so by having each user sample a key for a pseudorandom function (PRF) and derive keys as the PRF output. This allows for keys to appear uncorrelated to any colluding issuers, while allowing to prove (in zero-knowledge) that the keys are derived from the same PRF key.

For VRC issuers we provide cross-context unlinkability. Here, for a given context ctx, the VRC issuer computes a fresh key as the PRF output of ctx. This is proven when issuing the VRC, and since it is fixed for a context, VRC issuances within the same contexts are linkable, whereas across contexts they are not.

7.2.1 Components

We specify the components underlying our construction

- t (potentially distinct) Vouchable Credentials (Definition 9): $\{\mathsf{vCred}^{(i)} = (\mathsf{Init}^{(i)}, \mathsf{Issue}^{(i)}_{I,U}, \mathsf{Vouch}^{(i)}, \mathsf{Verify}^{(i)})\}_{i\in[t]}$
- A pseudorandom function (Section 3.4) $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$.
- A SE-NIZK (Definition 5) $\mathsf{NIZK}_{\mathsf{PRF}} = (\mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{PRF}}$.
- A SE-NIZK $\mathsf{NIZK}_{\mathsf{VRC}} = (\mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{VRC}}$.
- A SE-NIZK $\mathsf{NIZK}_{\mathsf{ShowVRC}} = (\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{ShowVRC}}$.

7.2.2 Proof of Knowledge Languages

We describe here the various languages that will be used in our protocol.

Proof of Knowledge of PRF evaluation. The user will generate a zkPoK for the language $\mathcal{L}_{\mathsf{PRF}}$ to establish ownership of the PRF key corresponding to the PRF evaluation i.e.

$$\mathcal{L}_{\mathsf{PRF}} = \{(x, y) \mid \exists \mathsf{K} \text{ s.t. } y = \mathsf{PRF.Eval}(\mathsf{K}, x)\}.$$

We assume here that it is efficient to test whether $(\mathsf{pk}, \mathsf{sk})$ is a valid publi-key secret-key pair.

Proof of Correct VRC generation. A VRC issuer generates a zkPoK for the language $\mathcal{L}_{\mathsf{VRC}}$, establishing that: (i) it dervied the issuer-specific key and context-specific key as the correct PRF evaluations using the same long term secret PRF key; (ii) there is a vouch on the statement st that verifies using the (hidden) issuer-specific key.

$$\begin{aligned}\mathcal{L}_{\mathsf{VRC}} = \Big\{&(\mathsf{st}, \mathsf{ipk}, \mathsf{att}, \mathsf{idx}, \mathsf{ctx}, k, k') \mid \exists \mathsf{K}, \pi, c \text{ s.t.}\\
&\mathsf{att}' = \mathsf{PRF.Eval}(\mathsf{K}, \mathsf{ipk})||\mathsf{att} \wedge \text{ //Correct issuer-specific key used in att}'\\
&k = \mathsf{PRF.Eval}(\mathsf{K}, \mathsf{ctx}) \wedge \text{ //Context-specific key derived correctly}\\
&\mathsf{idx}' = \mathsf{idx} \cup \{1\} \wedge \mathsf{st}' = \mathsf{st}||k' \wedge\\
&\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}', \mathsf{att}', \mathsf{idx}', \pi, c) = 1 \text{ //Vouch verifies using the issuer-specific key}\Big\}\end{aligned}$$

Proof of knowledge of VRC. A user who has collected many VRCs generates a zkPoK for the language $\mathcal{L}_{\mathsf{ShowVRC}}$, establishing that for claimed statements $\mathsf{st}_1, \ldots, \mathsf{st}_N$ and a predicate f: (i) the user holds valid VRCs issued with attributes $\mathsf{att}_1, \ldots, \mathsf{att}_N$ respectively; and (ii) $f(\mathsf{att}_1, \mathsf{st}_1, \ldots, \mathsf{att}_N, \mathsf{st}_N) = 1$.

We formally specify these constraints below, and note that the language is also parameterized by the issuer keys $\{\mathsf{ipk}^{(i)}\}_{i\in[t]}$ and the revocation list RL, but for ease of exposition we omit this in the language.

$$\begin{aligned}\mathcal{L}_{\mathsf{ShowVRC}} = \Big\{&(f, \mathsf{pk}_U, (\mathsf{st}_1, i_1, \mathsf{ctx}_1), \ldots, (\mathsf{st}_N, i_N, \mathsf{ctx}_N)) \mid \exists \mathsf{K}, \{\mathsf{att}_j, k_j, \mathsf{idx}_j, \pi_j\}_{j\in[N]} \text{ s.t.}\\
&\forall j \in [N], \mathsf{VerifyVRC}(\mathsf{ipk}^{(i_j)}, k'_j, \mathsf{st}_j, k_j, \mathsf{att}_j, \mathsf{idx}_j, \mathsf{ctx}, \pi_j) = 1 \wedge \text{ //Statements have a VRC}\\
&\forall j \in [N], k'_j = \mathsf{PRF.Eval}(\mathsf{K}, k_j) \wedge \text{ //VRC issued to keys owned by the same user}\\
&\mathsf{pk}_U = \mathsf{PRF.Eval}(\mathsf{K}, 0) \wedge \text{ //public key consistent with the VRC receiver keys}\\
&f(\mathsf{att}_1, \mathsf{st}_1, \mathsf{ctx}_1, k_1, \ldots, \mathsf{att}_N, \mathsf{ctx}_N, \mathsf{st}_N, k_N) = 1 \text{ //Statements and attributes satisfy predicate } f\Big\}\end{aligned}$$

7.2.3 Construction

The construction is specified in Fig. 3. We note that each user U also has a long term key pk_U, which can be derived from their PRF key K_U as $\mathsf{pk}_U = \mathsf{PRF.Eval}(K_U)$.

Proofs of Personhood - Unlinkable

Parameters: Security paramater 1^λ, number of credentials supported 1^t, number of attributes supported within each credential 1^ℓ, hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^\lambda$. A set of t issuers, $\{I^{(i)}\}_{i\in[t]}$.

$\underline{\mathsf{Setup}(1^\lambda, 1^\ell, 1^t)}$:

```
1. For each $i \in [t]$, $(\mathsf{ipk}^{(i)}, \mathsf{isk}^{(i)}) \leftarrow \mathsf{Init}^{(i)}(1^\lambda, 1^{\ell+1})$ //Additional for public key
2. $\mathsf{crs}_{\mathsf{PRF}} \leftarrow \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Setup}(1^\lambda)$
3. $\mathsf{crs}_{\mathsf{ShowVRC}} \leftarrow \mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Setup}(1^\lambda)$
4. Set $\mathsf{pp} = (\mathsf{crs}_{\mathsf{pk}}, \mathsf{crs}_{\mathsf{ShowVRC}}\{\mathsf{ipk}^{(i)}\}_{i\in[t]})$.
5. Ouput $(\mathsf{pp}, \{\mathsf{isk}^{(i)}\}_{i\in[t]})$.
```

$\underline{\mathsf{IssuePHC}(\langle I^{(i)}(\mathsf{pp}, \mathsf{isk}^{(i)}), U(\mathsf{pp}, \mathsf{ipk}^{(i)}, \mathsf{att}_U, \mathsf{K}_U, \mathsf{sk}_U)\rangle)}$:

```
$U \rightarrow I^{(i)}$: //User derives key and proves correct derivation
    1. $U$ computes $k_U^{(i)} = \mathsf{PRF.Eval}(\mathsf{K}_U, \mathsf{ipk}^{(i)})$.
    2. $U$ computes $\pi_{\mathsf{NIZK,PRF}} \leftarrow \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{PRF}}, (\mathsf{ipk}^{(i)}, k_U^{(i)}), \mathsf{K}_U)$
    3. Send $\pi_{\mathsf{NIZK,PRF}}$ along with $k_U^{(i)}$ and $\mathsf{att}_U$ to $I^{(i)}$.
$U \leftrightarrow I^{(i)}$ //User and Issuer run underlying issuance protocol with additional fixed attribute $k_U^{(i)}$
    1. $I^{(i)}$ outputs $\perp$ if $\mathsf{NIZK}_{\mathsf{pk}}.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_U, \pi_{\mathsf{NIZK,pk}}) = 0$.
    2. Set $\mathsf{att} = k_U^{(i)}||\mathsf{att}_U$
    3. $U$ and $I^{(i)}$ run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\mathsf{isk}^{(i)}, \mathsf{att}), (\mathsf{ipk}^{(i)}, \mathsf{att})\rangle$
    4. $U$ obtains $\mathsf{cred}_U^{(i)}$ at the end of the protocol.
```

$\underline{\mathsf{IssueVRC}(\mathsf{pp}, \mathsf{cred}_U^{(i)}, \mathsf{K}_U, \mathsf{att}_U, \mathsf{idx}, k_{U,U'}, \mathsf{st}, \mathsf{ctx})}$: //The VRC receiver samples a new key for each VRC received

```
1. Set $\mathsf{idx}' = \mathsf{idx} \cup \{1\}$ //Always reveal derived key during a VRC issuance
2. Set $\mathsf{st}' = \mathsf{st}||k_{U,U'}$. //Tie statement proven to $k_{U,U'}$
3. Set $\mathsf{att} = k_U^{(i)}||\mathsf{att}_U$.
4. Set $k_{U,\mathsf{ctx}} = \mathsf{PRF.Eval}(\mathsf{K}_U, \mathsf{ipk}||\mathsf{ctx})$. //The issuer generates a new context specific key
5. Compute $\pi_U^{(i)}, c_U^{(i)} \leftarrow \mathsf{Vouch}^{(i)}(\mathsf{st}', \mathsf{att}, \mathsf{idx}', \mathsf{cred}_U^{(i)})$
6. Set $\mathsf{x}_{\mathsf{VRC}} = (\mathsf{st}, \mathsf{ipk}^{(i)}, \mathsf{att}|_{\mathsf{idx}}, \mathsf{idx}, \mathsf{ctx}, k_{U,\mathsf{ctx}}, k_{U,U'})$ //Generate proof of $\mathcal{L}_{\mathsf{VRC}}$
7. Set $\mathsf{w}_{\mathsf{VRC}} = (\mathsf{K}_U, \pi_U^{(i)}, c_U^{(i)})$
8. Compute $\pi_{U,\mathsf{VRC}}^{(i)} \leftarrow \mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{VRC}}, \mathsf{x}_{\mathsf{VRC}}, \mathsf{w}_{\mathsf{VRC}})$.
9. Output $\pi_{U,\mathsf{VRC}}^{(i)}$, st, $\mathsf{ipk}^{(i)}$, $\mathsf{att}|_{\mathsf{idx}}$, idx, ctx, $k_{U,\mathsf{ctx}}$, $k_{U,U'}$.
```

$\underline{\mathsf{VerifyVRC}(\mathsf{pp}, \mathsf{ipk}^{(i)}, k_{U,U'}, \mathsf{st}, k_{U,\mathsf{ctx}}, \mathsf{att}, \mathsf{idx}, \mathsf{ctx}, \pi_{U,\mathsf{VRC}}^{(i)})}$:

```
1. Set Set $\mathsf{x}_{\mathsf{VRC}} = (\mathsf{st}, \mathsf{ipk}^{(i)}, \mathsf{att}|_{\mathsf{idx}}, \mathsf{idx}, \mathsf{ctx}, k_{U,\mathsf{ctx}}, k_{U,U'})$.
```

2. Output $\mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{VRC}}, \mathsf{x}_{\mathsf{VRC}}, \pi_{\mathsf{VRC}})$. //Verify VRC NIZK

$\mathsf{ShowVRC}(\mathsf{pp}, \mathsf{pk}_U, f, \mathsf{K}_U, \{\mathsf{ipk}^{(i_j)}, \mathsf{st}_j, k_j, \mathsf{att}_j, \mathsf{idx}_j, \pi_j, \mathsf{ctx}_j\}_{j\in[N]})$:

1. Set $\mathsf{x}_{\mathsf{ShowVRC}} = (f, \mathsf{pk}_U, (\mathsf{st}_1, i_1, \mathsf{ctx}_1), \ldots, (\mathsf{st}_N, i_N, \mathsf{ctx}_N))$ //Generate proof of $\mathcal{L}_{\mathsf{ShowVRC}}$
2. Set $\mathsf{w}_{\mathsf{ShowVRC}} = (\mathsf{K}_U, \{\mathsf{att}_j, k_j, \mathsf{idx}_j, \pi_j\}_{j\in[N]})$
3. Compute $\pi_{\mathsf{ShowVRC}} \leftarrow \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}, \mathsf{w}_{\mathsf{ShowVRC}})$.
4. Output $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$.

$\mathsf{VerifyShow}(\mathsf{pp}, \mathsf{pk}_U, f, (\mathsf{st}_1, i_1, \mathsf{ctx}_1), \ldots, (\mathsf{st}_N, i_N, \mathsf{ctx}_N), \pi_{\mathsf{ShowVRC}})$:

1. Set $\mathsf{x}_{\mathsf{ShowVRC}} = (f, \mathsf{pk}_U, (\mathsf{st}_1, i_1, \mathsf{ctx}_1), \ldots, (\mathsf{st}_N, i_N, \mathsf{ctx}_N))$.
2. Output $\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}, \pi_{\mathsf{ShowVRC}})$.

Figure 3: Proof of Personhood realizing $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$.

Theorem 7 (Realization of $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ with PAV Oracles). *Assume the security of the underlying vouchable-credential scheme (Definition 9), pseudorandom functions (Section 3.4) and the non-interactive zero-knowledge proof system (Definition 5). Then the proof-of-personhood system*

$$(\mathsf{Setup}, \mathsf{IssuePHC}, \mathsf{IssueVRC}, \mathsf{VerifyVRC}, \mathsf{ShowVRC}, \mathsf{VerifyShow})$$

above securely realizes the ideal functionality $\mathcal{F}_{\mathsf{PoP\text{-}user\text{-}unlinkability}}$ in the real/ideal world model with respect to the PAV oracle ensemble defined in Section 4.

7.2.4 Security Proof

Simulator for the unlinkable protocol. We now describe the simulation strategy for the unlinkable protocol. In particular, for every adversary $\mathcal{A}$ we construct a simulator $\mathcal{S}$ that interacts with $\mathcal{A}$ and $\mathcal{F}_{\mathsf{PoP}}$. $\mathcal{E}$ provides inputs to the honest parties, and the messages of $\mathcal{E}$. In this work it will suffice to think of $\mathcal{A}$ as relaying messages provided by $\mathcal{E}$, which may not follow the protocol specification, i.e., may be maliciously generated.

The simulation strategy is similar to the proof in Section 7.1.4. The main change in the description for $\mathcal{S}$ is noted below.

For honest parties, $\mathcal{S}$ never evaluates PRFs. Whenever unlinkable pseudonyms/keys are needed, $\mathcal{S}$ uses the random values already provided by $\mathcal{F}_{\mathsf{PoP}}$ (e.g., issuance pseudonyms and context keys) and simulates the corresponding NIZKs with the simulation trapdoor. For corrupted parties, when they send unlinkable NIZKs asserting PRF consistency, $\mathcal{S}$ extracts the underlying PRF key/witness from those proofs and checks consistency against the claimed derived keys.

Further, whenever $\mathcal{S}$ invokes $\mathcal{F}_{\mathsf{PoP}}$ on behalf of a corrupted user and $\mathcal{F}_{\mathsf{PoP}}$ samples a fresh pseudonym, $\mathcal{S}$ records the correspondence

$$(\mathsf{K}^*, \xi_{\mathsf{clear}}, \xi_{\mathsf{ideal}})$$

where K^* is the extracted PRF key identifying that corrupted user, ξ_{clear} is the key/pseudonym value sent in the clear by $\mathcal{A}$, and ξ_{ideal} is the pseudonym sampled by $\mathcal{F}_{\mathsf{PoP}}$.

For corrupted-party context-key usage, $\mathcal{S}$ similarly records

$$(\mathsf{K}^*, \mathsf{ctxt}, \eta_{\mathsf{clear}}, \eta_{\mathsf{ideal}})$$

where η_{ideal} is returned by $\mathcal{F}_{\mathsf{PoP}}$ on a ctxtKey query and η_{clear} is the corresponding value used by $\mathcal{A}$.

Setup. $\mathcal{S}$ samples the CRS for the NIZKs, along with simulation trapdoors. $\mathcal{S}$ also samples keys on behalf of honest issuers.

1. $\mathsf{crs}_{\mathsf{PRF}}, \mathsf{td}_{\mathsf{PRF}} \leftarrow \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Setup}(1^\lambda)$.
2. $\mathsf{crs}_{\mathsf{ShowVRC}}, \mathsf{td}_{\mathsf{ShowVRC}} \leftarrow \mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Setup}(1^\lambda)$.
3. $\mathsf{crs}_{\mathsf{VRC}}, \mathsf{td}_{\mathsf{VRC}} \leftarrow \mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Setup}(1^\lambda)$.
4. For each $I^{(i)} \notin C$, $(\mathsf{ipk}^{(i)}, \mathsf{isk}^{(i)}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
5. Initialize $\mathsf{CredList} \leftarrow \emptyset$ and $\mathsf{AttList} \leftarrow \emptyset$.
6. Initialize a mapping table $\mathsf{PseudoMap}$ for tuples $(\mathsf{K}^*, \xi_{\mathsf{clear}}, \xi_{\mathsf{ideal}})$.
7. Initialize a mapping table CtxMap for tuples $(\mathsf{K}^*, \mathsf{ctxt}, \eta_{\mathsf{clear}}, \eta_{\mathsf{ideal}})$.

Credential Issuance. For credential issuance, $\mathcal{S}$ simulates interaction only when exactly one party between issuer and user is corrupted.

- If the credential recipient user $U \in C$:
  1. Receive from $\mathcal{A}$ (on behalf of U) $\pi_{\mathsf{NIZK,PRF}}$, ξ, and att_U intended for honest issuer $I^{(i)}$, where ξ is the issuer-facing unlinkable key/pseudonym.
  2. Output $\perp$ if $\mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{PRF}}, \xi, \pi_{\mathsf{NIZK,PRF}}) = 0$.
  3. Run $\mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{Ext}$ on $\pi_{\mathsf{NIZK,PRF}}$ to extract the PRF key K^* and check that it is consistent with ξ (and the statement expected in this issuance step). If this fails, output $\perp_{\mathsf{NIZK}}$ and abort.
  4. Send $(\mathsf{credIssuance}, I^{(i)}, \mathsf{att}_U)$ to $\mathcal{F}_{\mathsf{PoP}}$.
  5. Forward $\mathscr{U}$ from U to $O_\lambda^{(i)}$ unchanged.
  6. If $\mathcal{F}_{\mathsf{PoP}}$ sends $(\mathsf{credApproved}, \xi_{\mathsf{ideal}})$, record $(\mathsf{K}^*, \xi, \xi_{\mathsf{ideal}})$ in $\mathsf{PseudoMap}$, run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\mathsf{isk}^{(i)}, \mathsf{att}_U), (\cdot,\cdot)\rangle$ with U, behaving honestly on behalf of $I^{(i)}$, and add $(U, I^{(i)}, \xi_{\mathsf{ideal}}, \mathsf{att}_U)$ to $\mathsf{CredList}$. //The psuedonym chosen by the functionality ξ_{ideal} is not revealed to $\mathcal{A}$
- If the credential issuer $I^{(i)} \in C$:
  1. Receive $(\mathsf{credIssuance}, \xi, \mathsf{att})$ from $\mathcal{F}_{\mathsf{PoP}}$, where ξ is the unlinkable pseudonym provided by the functionality.
  2. Simulate $\pi_{\mathsf{NIZK,PRF}} \leftarrow \mathsf{NIZK}_{\mathsf{PRF}}.\mathsf{SimProve}(\mathsf{crs}_{\mathsf{PRF}}, \mathsf{td}_{\mathsf{PRF}}, \xi)$, and send $\pi_{\mathsf{NIZK,PRF}}$, ξ, and att to $\mathcal{A}$. //Use the pseudonym sent by the ideal functionality and simulate the NIZK explaining ownership
  3. Forward att' from $\mathcal{A}$ to $O_\lambda^{(i)}$ and return the response.
  4. Run $\mathsf{Issue}^{(i)}_{I^{(i)},U}\langle(\cdot,\cdot), (\mathsf{ipk}^{(i)}, \mathsf{att})\rangle$ with $I^{(i)}$, behaving honestly on behalf of U.

VRC Issuance. $\mathcal{S}$ simulates the VRC issuance interaction between honest and corrupted users and maintains attestation bookkeeping.

- If the VRC recipient user $U \in C$:
  1. Receive from $\mathcal{A}$ (on behalf of U) $(\eta_{\mathsf{clear}}, \mathsf{ctxt}, \mathsf{st})$ for VRC issuance by η_{ctxt}. //$\mathcal{A}$ does not provide proof of consistency for η_{clear} as it is not checked during VRC issuance.
  2. If $\mathcal{S}$ has not sent η_{ctxt}, on behalf of a honest party, to $\mathcal{A}$ return $\perp$. //Indicates no honest party with the context key.
  3. Send $(\mathsf{relAttestIssuance}, \eta_{\mathsf{ctxt}}, \mathsf{ctxt}, \mathsf{st})$ to $\mathcal{F}_{\mathsf{PoP}}$.
  4. On receiving $(\mathsf{relAttestApproved}, \xi_{\mathsf{ideal}}, \mathsf{att}', \mathsf{idx}, I^{(i)}, \mathsf{id})$ from $\mathcal{F}_{\mathsf{PoP}}$, generate vouch π as $\mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{SimProve}(\mathsf{crs}_{\mathsf{VRC}}, \mathsf{td}_{\mathsf{VRC}}, \mathsf{st}, \mathsf{ipk}^{(i)}, \mathsf{att}', \mathsf{idx}, \mathsf{ctxt}, \eta_{\mathsf{ctxt}}, \eta_{\mathsf{clear}})$.
  5. Add $(\mathsf{id}, \xi_{\mathsf{ideal}}, \eta_{\mathsf{ctxt}}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to $\mathsf{AttList}$.
  6. Send π, $\mathsf{ipk}^{(i)}$, st, $\mathsf{idx} \cup \{1\}$, and $\eta_{\mathsf{clear}}||\mathsf{att}'$ to $\mathcal{A}$.
- If the VRC issuer user $U \in C$:
  1. Receive $(\mathsf{relAttestIssuance}, \xi, \mathsf{st})$ from $\mathcal{F}_{\mathsf{PoP}}$, where ξ is the requester pseudonym.
  2. Send $(\mathsf{relAttestIssuance}, \xi, \mathsf{st})$ to the corrupted attestation issuer (via $\mathcal{A}$).
  3. On receiving π_{VRC}, $\mathsf{ipk}^{(i)}$, st, idx, and $\xi_{\mathsf{iss}}||\mathsf{att}'$ from $\mathcal{A}$, set $\mathsf{idx}' = \mathsf{idx} \cup \{1\}$ and $\mathsf{st}' = \mathsf{st}$, then check
  $$\mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{VRC}}, \mathsf{x}_{\mathsf{VRC}}, \pi_{\mathsf{VRC}}) = 1$$
  for the corresponding statement $\mathsf{x}_{\mathsf{VRC}}$.

4. Run $\mathsf{NIZK}_{\mathsf{VRC}}.\mathsf{Ext}$ on π_{VRC} to extract witness components, including the underlying vouch object (π, c) and the full attribute vector data used in the statement.
5. Check that the extracted witness is valid for $\mathsf{x}_{\mathsf{VRC}}$ in $\mathcal{L}_{\mathsf{VRC}}$; otherwise abort with $\bot_{\mathsf{NIZK}}$.
6. Run Ext of the vouchable credential on the extracted vouch object (π, c) to obtain (att^*, r).
7. If vouch extraction fails or $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$, abort with $\bot_{\mathsf{VouchExtract}}$.
8. If $I^{(i)} \in C$, register the underlying PHC with $\mathcal{F}_{\mathsf{PoP}}$ by internally simulating issuance for both parties:
   (a) Send $(\mathsf{credIssuance}, I^{(i)}, \mathsf{att}^*)$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of U.
   (b) Receive $(\mathsf{credIssuance}, \xi_{\mathsf{iss}}, \mathsf{att}^*)$ from $\mathcal{F}_{\mathsf{PoP}}$ on behalf of $I^{(i)}$.
   (c) Send $(\mathsf{credApproved}, \xi_{\mathsf{iss}})$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of $I^{(i)}$.
   (d) Add $(U, I^{(i)}, \xi_{\mathsf{iss}}, \mathsf{att}^*)$ to $\mathsf{CredList}$.
9. If $I^{(i)} \notin C$, check whether $(U, I^{(i)}, \xi_{\mathsf{iss}}, \mathsf{att}^*) \in \mathsf{CredList}$. If not, abort with $\bot_{\mathsf{PHCforgery}}$.
10. Send $(\mathsf{approveRelAttest}, \xi, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to $\mathcal{F}_{\mathsf{PoP}}$.
11. On receiving $(\mathsf{relAttestApproved}, \mathsf{id})$ from $\mathcal{F}_{\mathsf{PoP}}$, add $(\mathsf{id}, \xi, \xi_{\mathsf{iss}}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, I^{(i)})$ to $\mathsf{AttList}$.

Showing. $\mathcal{S}$ simulates showing between honest and corrupted users.

- If the show recipient user $U \in C$:
  1. Receive $(\mathsf{showCred}, V, f, (\mathsf{st}_1, I^{(i_1)}, \ldots, \mathsf{st}_N, I^{(i_N)}))$ from $\mathcal{F}_{\mathsf{PoP}}$.
  2. Set
  $$\mathsf{x}_{\mathsf{ShowVRC}} = (\mathsf{pk}_U, f, (\mathsf{st}_1, i_1), \ldots, (\mathsf{st}_N, i_N)).$$
  3. Simulate π_{ShowVRC} and send $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$ to $\mathcal{A}$.
  4. In simulated showing for honest users, use the pseudonyms/keys already supplied by $\mathcal{F}_{\mathsf{PoP}}$ and simulate the show NIZK; $\mathcal{S}$ does not compute any honest-user PRF values.
- If the show issuer user $U \in C$:
  1. Receive $\pi_{\mathsf{ShowVRC}}, \mathsf{x}_{\mathsf{ShowVRC}}$ from $\mathcal{A}$.
  2. Abort if π_{ShowVRC} does not verify.
  3. Run $\mathsf{NIZK}_{\mathsf{ShowVRC}}.\mathsf{Ext}$ to extract
  $$\mathsf{w}_{\mathsf{ShowVRC}} = \{\mathsf{att}_j, \xi_j, \mathsf{idx}_j, \pi_j\}_{j \in [N]},$$
  and output $\bot_{\mathsf{NIZK}}$ if $\mathsf{w}_{\mathsf{ShowVRC}}$ is not a valid witness for $\mathsf{x}_{\mathsf{ShowVRC}}$ in $\mathcal{L}_{\mathsf{ShowVRC}}$.
  4. For each $j \in [N]$, let $(\mathsf{att}_j, \xi_j, \mathsf{idx}_j, \pi_j)$ be the extracted tuple corresponding to issuer $I^{(i_j)}$, and let ξ_{recv} denote the receiver handle bound by the extracted show statement (in the linkable showing variant this may equal pk_U). Perform case analysis:
     (a) If $I^{(i_j)} \notin C$ and the user corresponding to ξ_j is honest, check for a matching attestation record in $\mathsf{AttList}$ for $(\xi_{\mathsf{recv}}, \xi_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$. If no such record exists, abort with $\bot_{\mathsf{VRCforgery}}$.
     (b) If $I^{(i_j)} \notin C$ but the user corresponding to ξ_j is corrupted, check whether the extracted underlying credential corresponds to a PHC previously issued and recorded in $\mathsf{CredList}$. If not, abort with $\bot_{\mathsf{PHCforgery}}$. Otherwise invoke $\mathcal{F}_{\mathsf{PoP}}$ to issue the corresponding relationship attestation, receive identifier id_j, and add $(\mathsf{id}_j, \xi_{\mathsf{recv}}, \xi_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$ to $\mathsf{AttList}$.
     (c) If $I^{(i_j)} \in C$ but the user corresponding to ξ_j is honest, check for a matching attestation record in $\mathsf{AttList}$ for $(\xi_{\mathsf{recv}}, \xi_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$. If none exists, abort with $\bot_{\mathsf{VRCforgery}}$.
     (d) If both $I^{(i_j)} \in C$ and the user corresponding to ξ_j is corrupted, simulate the corresponding relationship-attestation issuance in $\mathcal{F}_{\mathsf{PoP}}$ on behalf of corrupted parties, receive identifier id_j, and add $(\mathsf{id}_j, \xi_{\mathsf{recv}}, \xi_j, \mathsf{st}_j, \mathsf{att}_j, \mathsf{idx}_j, I^{(i_j)})$ to $\mathsf{AttList}$.
  5. Let $\mathsf{id}_1, \ldots, \mathsf{id}_N$ be the attestation identifiers returned by $\mathcal{F}_{\mathsf{PoP}}$ (or recorded earlier) from the checks above. Send $(\mathsf{showCred}, V, f, (\mathsf{id}_1, \ldots, \mathsf{id}_N))$ to $\mathcal{F}_{\mathsf{PoP}}$ on behalf of the corrupted user.

6. When processing corrupted-user shows, extract the PRF witness from the corresponding NIZK and check that all asserted PRF relations in the extracted tuple are satisfied and consistent with $\mathsf{PseudoMap}/\mathsf{CtxMap}$; otherwise abort with $\bot_{\mathsf{NIZK}}$.

Hybrids. We now spell out the hybrid sequence in the same style as the base proof.

Remark 5. *In the unlinkable construction, the vouch object (including the commitment) appears only inside the witness of the unlinkable NIZK and is never exposed as a standalone transcript item to the adversary. Hence the proof below does not require separate vouch non-malleability or vouch simulatability hybrids. In this setting, weaker assumptions may suffice, provided extracted-witness consistency and the remaining soundness checks are preserved. In particular, our generic transformation (Appendix C) upgrading credentials to vouchable credentials makes use of NIZKs to obtain vouch non-malleability and vouch simulatability. Here, because the unlinkable construction already explicitly uses NIZKs, those properties need not be introduced as separate assumptions/hybrid steps.*

Hyb_0: *Real world.* This is the real execution of the unlinkable protocol.

Hyb_1: *Simulated NIZKs for honest parties.* Replace all honest-party NIZKs by simulated proofs using trapdoors.

Claim 9.

$$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$$

Proof. Indistinguishability follows from the (multi-theorem) zero-knowledge property of the NIZK systems; the reduction is exactly as in Claim 1. □

Hyb_2: *Extraction from NIZKs.* Whenever a corrupted party sends an accepting NIZK proof, run extraction and abort on invalid/inconsistent extraction.

Claim 10.

$$\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$$

Proof. Indistinguishability follows from weak simulation-extractability of the NIZK systems; the same reduction applies as in Claim 2. □

Hyb_3: *Recording issued PHCs and VRCs.* Maintain the same local bookkeeping as in the base proof:

$$\mathsf{CredList} \text{ for PHCs}, \qquad \mathsf{AttList} \text{ for attestations/VRC metadata (including } \mathsf{id}).$$

Claim 11.

$$\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$$

Proof. This is an identical-distribution step (pure local bookkeeping with no message change), exactly as in Claim 3. □

Hyb_4: *Abort on vouch-extraction inconsistency.* Whenever $\mathcal{S}$ first extracts a valid unlinkable-NIZK witness and then runs the vouch extractor on the extracted vouch object, abort with $\bot_{\mathsf{VouchExtract}}$ if vouch extraction fails or if the extracted attributes are inconsistent with the revealed projection.

Claim 12.

$$\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$$

Proof. Indistinguishability follows from vouch extractability, with the same reduction pattern as Claim 4. □

Hyb_5: *Abort on forged PHCs.* Add the same PHC-consistency abort checks as in the base proof.

Claim 13.

$$\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$$

Proof. Indistinguishability follows from unforgeability of the underlying vouchable credential scheme, with the same reduction structure as Claim 5. □

Hyb_6: *Replace honest PRF-derived keys with ideal-sampled random keys.* For honest users, replace real PRF-derived unlinkable keys in issuance/showing transcripts by the random pseudonyms/context keys provided by $\mathcal{F}_{\mathsf{PoP}}$, while simulating the corresponding NIZKs.

Claim 14.

$$\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$$

Proof. This is the unlinkable-specific hop. We reduce to PRF security via a standard single-hop replacement argument over polynomially many honest-party PRF evaluations.

Assume there exists a distinguisher for this hybrid transition with non-negligible advantage. Then there exists an index of an honest PRF call such that replacing only that call changes the view with non-negligible advantage. We build $\mathcal{B}_{\mathsf{PRF}}$ against PRF pseudorandomness:

$\underline{\mathcal{B}^{\mathsf{O}}_{\mathsf{PRF}}(\cdot)}$

1. Emulate the unlinkable execution up to this hop, including all prior checks/hybrids.
2. For all honest PRF evaluations except the selected one, answer consistently using real PRF evaluations.
3. For the selected evaluation, query oracle O on the corresponding input and use the returned value in the emulation.
4. Output the final bit of the emulated environment.

If O is a real PRF oracle, the emulation is distributed as the previous sub-hybrid; if O is uniform random, it is distributed as the next sub-hybrid. A non-negligible distinguisher therefore breaks PRF security. Summing over polynomially many calls yields $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$. □

Hyb_7: *Ideal world.* This final hybrid is exactly the unlinkable simulator execution with all above transformations applied.

Claim 15.

$$\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$$

Proof. This is an identical-distribution step after all prior transformations (simulated/extracted NIZKs, bookkeeping, vouch-extraction consistency, PHC-consistency checks, and PRF replacement). □

8 Blackbox Vouchable Credentials

In this section, we describe our blackbox construction of vouchable credentials.

We begin by recalling the [Gro15] structure-preserving signature scheme that signs $N = mn$ $\mathbb{G}_2$ elements with a verification key of size m group elements and produces a signature of $n+2$ group elements. Verification of the signature requires $n+1$ pairing product equations. For clarity, we focus on the parameters of interest ($m = 1$ and $n = 2$) i.e., signing two group elements where the public key is a single group element target EUF-CMA security (randomizable signatures).

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, with generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$.

$\mathsf{Setup}(1^\lambda)$: Sample $y_1, y_2 \leftarrow\$ \mathbb{F}$ and output $\mathsf{p} = (g, h, Y_1 = h^{y_1}, Y_2 = h^{y_2})$.

$\mathsf{KeyGen}(\mathsf{p})$: Sample $v \leftarrow\$ \mathbb{F}$ and output $\mathsf{vk} := V = g^v \in \mathbb{G}_1$ and $\mathsf{sk} := v$.

$\mathsf{Sign}(\mathsf{sk}, (M_1, M_2))$: To sign a message $(M_1, M_2) \in \mathbb{G}_2^2$:

- Sample $r \leftarrow\$ \mathbb{F}^*$ and compute $R := g^r$.
- Compute $S := (Y_1 \cdot h^v)^{\frac{1}{r}}$.
- Compute $T_1 := (M_1 \cdot Y_1^v)^{\frac{1}{r}}$ and $T_2 := (M_2 \cdot Y_2^v)^{\frac{1}{r}}$.
- Output the signature $\sigma = (R, S, T_1, T_2) \in \mathbb{G}_1 \times \mathbb{G}_2^3$.

$\mathsf{Verify}(\mathsf{vk}, (M_1, M_2), \sigma)$: Parse $\mathsf{vk} = V$ and $\sigma = (R, S, T_1, T_2)$. Accept if

- $R \in \mathbb{G}_1$ and $S, T_1, T_2 \in \mathbb{G}_2$
- $e(R, S) = e(g, Y_1) \cdot e(V, h)$
- $e(R, T_1) = e(g, M_1) \cdot e(V, Y_1)$
- $e(R, T_2) = e(g, M_2) \cdot e(V, Y_2)$

[Gro15, Theorem 1] showed that the above signature scheme is EUF-CMA secure in the generic group model.

Remark 6. *By symmetry of the bilinear map, we can "flip" the scheme to have verification keys in $\mathbb{G}_2$ and sign messages in $\mathbb{G}_1$. The public parameters become $W_1 = g^{w_1}, W_2 = g^{w_2} \in \mathbb{G}_1$, and signature components are $R \in \mathbb{G}_2$ and $S, T_1, T_2 \in \mathbb{G}_1$. Security follows by an identical argument.*

8.1 Vouchable Credential Instantiation

We are now ready to present our Vouchable Credential scheme using the [Gro15] structure-preserving signature scheme. At a high-level, the issuer's public key is an SPS verification key and the credential is a signature on the user's public key and an index-position commitment to their attributes. To produce a vouch, the user presents their credential together with a signature (using their own key) on the statement and the revealed index set.

Construction 1. *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, with generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$.*

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{crs}, \mathsf{td})$:

- *Sample generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$.*
- *Sample $\tau, \eta \leftarrow\$ \mathbb{F}$ and compute $\{g^{\tau^i}\}_{i \in [0,\ell]}$, g^η, $\{h^{\tau^i}\}_{i \in [0,\ell]}$, and h^η. //KZG CRS*
- *Sample $y_1, y_2 \leftarrow\$ \mathbb{F}$ and compute $Y_1 = h^{y_1}, Y_2 = h^{y_2}$. //Issuer SPS params*
- *Sample $w_1, w_2 \leftarrow\$ \mathbb{F}$ and compute $W_1 = g^{w_1}, W_2 = g^{w_2}$. //User SPS params (flipped)*
- *Output $\mathsf{pp} = (g, h, Y_1, Y_2, W_1, W_2)$, $\mathsf{crs} = (\{g^{\tau^i}\}, \{h^{\tau^i}\}_{i \in [0,\ell]}, g^\eta, h^\eta)$ and $\mathsf{td} = (\tau, \eta)$.*

$\mathsf{Init}(1^\lambda, 1^\ell) \to (\mathsf{isk}, \mathsf{ipk})$: *// [Gro15] SPS KeyGen*

- *I samples a secret key $\mathsf{isk} := v_I \leftarrow\$ \mathbb{F}$ and outputs $\mathsf{ipk} := V_I = g^{v_I} \in \mathbb{G}_1$.*

$\mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle \to \mathsf{cred}/\bot$: *//All attributes are revealed*

- *U samples a secret key $v_U \leftarrow\$ \mathbb{F}$ and computes their public key $V_U := h^{v_U} \in \mathbb{G}_2$. // [Gro15] SPS KeyGen*
- *U samples $r_{\mathsf{att}} \leftarrow\$ \mathbb{F}$.*
- *U and I define the unique degree-$< \ell$ polynomial*

$$f_{\mathsf{att}}(X) \text{ such that } f_{\mathsf{att}}(i) = \mathsf{att}[i] \text{ for all } i \in [\ell]$$

  and U computes a Hiding KZG commitment as $\mathsf{com}_{\mathsf{att}} = h^{f_{\mathsf{att}}(\tau)} \cdot (h^{\eta})^{r_{\mathsf{att}}} \in \mathbb{G}_2$.
- *U proves knowledge of v_U and r_{att}:*

$$\pi_U = \{(v_U, r_{\mathsf{att}}) \mid V_U = h^{v_U} \wedge \mathsf{com}_{\mathsf{att}} = h^{f_{\mathsf{att}}(\tau)} \cdot (h^{\eta})^{r_{\mathsf{att}}}\}$$

  The proof must be straight-line extractable and can be instantiated using Σ-protocols in the GGM.
- *U sends $(V_U, \mathsf{com}_{\mathsf{att}}, \pi_U, \mathsf{att})$ to I.*
- *I aborts if the proof π_U is invalid.*
- *I signs the pair $(V_U, \mathsf{com}_{\mathsf{att}}) \in \mathbb{G}_2^2$ using [Gro15] and outputs the credential signature:*

$$\sigma_{\mathsf{cred}} = (R_{\mathsf{cred}}, S_{\mathsf{cred}}, T_{\mathsf{cred},1}, T_{\mathsf{cred},2}) \leftarrow \mathsf{Sign}(v_I, (V_U, \mathsf{com}_{\mathsf{att}}))$$

- *U outputs $\mathsf{cred} = (V_U, \mathsf{com}_{\mathsf{att}}, \sigma_{\mathsf{cred}}, v_U, r_{\mathsf{att}})$ as their credential.*

$\mathsf{Vouch}(\mathsf{st}, \mathsf{att}, \mathsf{idx}, \mathsf{cred}) \rightarrow (c, \pi)$: *To vouch for a statement* st*:*

- *Parse $\mathsf{cred} = (V_U, \mathsf{com}_{\mathsf{att}}, \sigma_{\mathsf{cred}}, v_U, r_{\mathsf{att}})$.*
- *U computes the Hiding KZG opening proof (π^1, π^2) for attributes in idx:*

$$\pi^1 = (g^{\eta})^s \cdot g^{q_{\mathsf{idx}}(\tau)}, \quad \pi^2 = g^{r_{\mathsf{att}}} \cdot (g^{u_{\mathsf{idx}}(\tau)})^{-1} \cdot (g^{z_{\mathsf{idx}}(\tau)})^{-s}$$

  where

$$z_{\mathsf{idx}}(X) = \prod_{i \in \mathsf{idx}} (X - i), \quad u_{\mathsf{idx}}(X) \text{ is degree-} < |\mathsf{idx}| \text{ with } u_{\mathsf{idx}}(i) = \mathsf{att}[i]\ \forall i \in \mathsf{idx},$$

$$q_{\mathsf{idx}}(X) = \frac{f_{\mathsf{att}}(X) - u_{\mathsf{idx}}(X)}{z_{\mathsf{idx}}(X)},$$

  and $s \leftarrow\$ \mathbb{F}$ is a random blinding factor.
- *U signs the pair $(g^{\mathsf{st}}, g^{z_{\mathsf{idx}}(\tau)}) \in \mathbb{G}_1^2$ using [Gro15] and outputs the vouch signature:*

$$\sigma_{\mathsf{st}} = (R_{\mathsf{st}}, S_{\mathsf{st}}, T_{\mathsf{st},1}, T_{\mathsf{st},2}) \leftarrow \mathsf{Sign}(v_U, (g^{\mathsf{st}}, g^{z_{\mathsf{idx}}(\tau)}))$$

Output $c = \mathsf{com}_{\mathsf{att}}$ and $\pi = (\sigma_{\mathsf{st}}, V_U, \sigma_{\mathsf{cred}}, \pi^1, \pi^2)$.

$\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) \rightarrow 0/1$: *Parse $\pi = (\sigma_{\mathsf{st}}, V_U, \sigma_{\mathsf{cred}}, \pi^1, \pi^2)$ and set $\mathsf{com}_{\mathsf{att}} = c$.*

- *Parse $\sigma_{\mathsf{cred}} = (R_{\mathsf{cred}}, S_{\mathsf{cred}}, T_{\mathsf{cred},1}, T_{\mathsf{cred},2})$ and verify the credential signature:*
  - $e(R_{\mathsf{cred}}, S_{\mathsf{cred}}) = e(g, Y_1) \cdot e(V_I, h)$
  - $e(R_{\mathsf{cred}}, T_{\mathsf{cred},1}) = e(g, V_U) \cdot e(V_I, Y_1)$
  - $e(R_{\mathsf{cred}}, T_{\mathsf{cred},2}) = e(g, \mathsf{com}_{\mathsf{att}}) \cdot e(V_I, Y_2)$
- *Verify the Hiding KZG batch proof:*
  - *Compute*

$$z_{\mathsf{idx}}(X) = \prod_{i \in \mathsf{idx}} (X - i)$$

    and the unique degree-$< |\mathsf{idx}|$ polynomial $u'_{\mathsf{idx}}(X)$ such that $u'_{\mathsf{idx}}(i) = \mathsf{att}'[i]$ for all $i \in \mathsf{idx}$.
  - *Check*

$$e\left(g, \mathsf{com}_{\mathsf{att}}/h^{u'_{\mathsf{idx}}(\tau)}\right) = e(\pi^1, h^{z_{\mathsf{idx}}(\tau)}) \cdot e(\pi^2, h^{\eta}).$$

- *Parse $\sigma_{\mathsf{st}} = (R_{\mathsf{st}}, S_{\mathsf{st}}, T_{\mathsf{st},1}, T_{\mathsf{st},2})$ and verify the flipped vouch signature:*
  - $e(S_{\mathsf{st}}, R_{\mathsf{st}}) = e(W_1, h) \cdot e(g, V_U)$
  - $e(T_{\mathsf{st},1}, R_{\mathsf{st}}) = e(g^{\mathsf{st}}, h) \cdot e(W_1, V_U)$
  - $e(T_{\mathsf{st},2}, R_{\mathsf{st}}) = e(g^{z_{\mathsf{idx}}(\tau)}, h) \cdot e(W_2, V_U)$

Output 1 if all checks pass, else output 0.

Remark 7. *Given that the vouch verification algorithm only involves pairing product equations where all witness terms are group elements, we can prove* knowledge *of a valid vouch using Groth-Sahai Proofs [GS08].*

8.2 Security Proof

We now prove that Construction 1 is a secure Vouchable Credential scheme.

Correctness. Correctness follows by inspection but we provide intuition for the construction below which will also aid in understanding the security proof. During Issue, the U samples a secret key v_U and computes their public key $V_U = h^{v_U} \in \mathbb{G}_2$. Both parties define the unique polynomial f_{att} with $f_{\mathsf{att}}(i) = \mathsf{att}[i]$ for all $i \in [\ell]$, and compute a Hiding KZG commitment $\mathsf{com}_{\mathsf{att}} = h^{f_{\mathsf{att}}(\tau)} \cdot (h^{\eta})^{r_{\mathsf{att}}} \in \mathbb{G}_2$. I signs the pair $(V_U, \mathsf{com}_{\mathsf{att}}) \in \mathbb{G}_2^2$ using [Gro15] to create the credential. To vouch for a statement st using attributes at positions idx, the user computes a Hiding KZG batch opening proof of the evaluations $\{\mathsf{att}[i]\}_{i \in \mathsf{idx}}$. The user then signs $(\mathsf{st}, g^{z_{\mathsf{idx}}(\tau)})$. The verification checks the credential signature, the user's signature on the statement and index set, and the KZG opening proof.

Lemma 1 (Unforgeability). *Construction 1 satisfies unforgeability against generic adversaries.*

Proof. We construct an extractor $\mathsf{Ext}(\mathsf{ipk}, z)$ that runs the adversary $\mathcal{A}$ and simulates the honest issuer oracle as follows:

Setup: Ext initializes an empty list L. It generates parameters $(\mathsf{pp}, \mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell})$ and issuer keys $(\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathsf{Init}(1^{\lambda}, 1^{\ell})$. It then invokes $\mathcal{A}^{O_{\mathsf{Issue}}}(\mathsf{ipk}, z)$.

Simulate O_{Issue}: When $\mathcal{A}$ initiates an Issue query for attributes att:

- $\mathcal{A}$ sends a message $(V_U, \mathsf{com}_{\mathsf{att}}, \pi_U, \mathsf{att})$.
- Ext, running the knowledge extractor for π_U, extracts the witnesses (v_U, r_{att}) such that $\mathsf{com}_{\mathsf{att}} = h^{f_{\mathsf{att}}(\tau)} \cdot (h^{\eta})^{r_{\mathsf{att}}}$.
- Ext stores the tuple $(c = \mathsf{com}_{\mathsf{att}}, \mathsf{att}, r_{\mathsf{att}})$ in the list L.
- Ext completes the rest of the issuance protocol honestly using isk.

Output: Eventually $\mathcal{A}$ outputs a forgery $(\mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$. Ext searches L for an entry where the stored commitment matches c.

- If found, let the entry be $(c, \mathsf{att}, r_{\mathsf{att}})$. Ext outputs $(\mathsf{att}^*, r^*) = (\mathsf{att}, r_{\mathsf{att}})$.
- Otherwise, if $c \notin L$, Ext outputs $\perp$.

We now analyze the probability that $\mathcal{A}$ wins the unforgeability game. $\mathcal{A}$ wins if $\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) = 1$ AND one of the following failure conditions holds:

1. $\mathsf{com}(\mathsf{att}^*; r^*) \neq c$
2. $\mathsf{att}^* \notin Q_{\mathsf{attr}}$
3. $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$

There are now two main cases for the adversary's output c:

Case 1: Uncertified Credential ($c \notin L$). If the adversary outputs a commitment c that is not in the list L (meaning it was not generated during a queried Issue session), then Ext fails (outputs $\perp$). However, for the vouch to verify, $\sigma_{\mathsf{cred}} = (R_{\mathsf{cred}}, S_{\mathsf{cred}}, T_{\mathsf{cred},1}, T_{\mathsf{cred},2})$ must be a valid signature on (V_U, c) under the issuer's key. Since c was never signed by the honest issuer (because all signed commitments are added to L), this constitutes a forgery against the SPS scheme [Gro15]. Thus, this case occurs with negligible probability.

Case 2: Certified Credential ($c \in L$). If $c \in L$, then Ext outputs (att^*, r^*) from the stored entry. Since the honest issuer verifies the proof of knowledge π_U before signing the credential, the soundness of the proof system implies that the extracted witnesses must satisfy the relation $\mathsf{com}_{\mathsf{att}} = h^{f_{\mathsf{att}^*}(\tau)} \cdot (h^{\eta})^{r^*}$ (except with negligible probability). Thus, $c = \mathsf{com}(\mathsf{att}^*; r^*)$ holds. If $\mathcal{A}$ wins, it must be that $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$. Let $u'_{\mathsf{idx}}(X)$ be the unique degree-$<|\mathsf{idx}|$ polynomial such that $u'_{\mathsf{idx}}(i) = \mathsf{att}'[i]$ for all $i \in \mathsf{idx}$, and let $z_{\mathsf{idx}}(X) = \prod_{i \in \mathsf{idx}}(X - i)$. The verification of the vouch checks:

$$e\left(g, c/h^{u'_{\mathsf{idx}}(\tau)}\right) = e(\pi^1, h^{z_{\mathsf{idx}}(\tau)}) \cdot e(\pi^2, h^{\eta}).$$

Since c already commits to $f_{\mathsf{att}^*}(X)$, KZG binding implies that any accepting opening at points idx must return the committed values at those points, i.e., $u'_{\mathsf{idx}}(i) = f_{\mathsf{att}^*}(i) = \mathsf{att}^*[i]$ for every $i \in \mathsf{idx}$. Therefore, $\mathsf{att}' = \mathsf{att}^*|_{\mathsf{idx}}$, a contradiction. If $\mathsf{att}^*|_{\mathsf{idx}} \neq \mathsf{att}'$, the adversary has broken the computational binding property of the KZG polynomial commitment scheme.

Hence, the probability of $\mathcal{A}$ winning the unforgeability game is negligible. □

Lemma 2 (Vouch Non-Malleability). *Construction 1 satisfies vouch non-malleability against generic adversaries.*

Proof. Suppose for contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks vouch non-malleability with non-negligible probability ϵ. That is, $\mathcal{A}$ chooses attributes att, an honest credential $\mathsf{cred} = (V_U, \mathsf{com}_{\mathsf{att}}, \sigma_{\mathsf{cred}}, v_U, r_{\mathsf{att}})$ is issued for att, and $\mathcal{A}$ is given oracle access to $\mathcal{O}_{\mathsf{Vouch}}(\cdot, \mathsf{att}, \cdot, \mathsf{cred})$. Let Q denote the set of queries $(\mathsf{st}, \mathsf{idx}, c)$ made by $\mathcal{A}$ to the oracle, where $c = \mathsf{com}_{\mathsf{att}}$ is fixed for the honest credential. Eventually, $\mathcal{A}$ outputs $(\pi, c', \mathsf{st}', \mathsf{idx}')$ such that $\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}', \mathsf{att}|_{\mathsf{idx}'}, \mathsf{idx}', \pi, c') = 1$ but $(\mathsf{st}', \mathsf{idx}', c') \notin Q$.

The adversary's forged vouch π parses as:

$$\pi = (\sigma^*_{\mathsf{st}}, V^*_U, \sigma^*_{\mathsf{cred}}, \pi^{1*}, \pi^{2*})$$

We analyze all possible cases and show that each leads to a contradiction.

Case 1: Modified Credential Data ($(V^*_U, c') \neq (V_U, \mathsf{com}_{\mathsf{att}})$). If the adversary outputs a credential tuple such that $(V^*_U, c') \neq (V_U, \mathsf{com}_{\mathsf{att}})$, then the verification of σ^*_{cred} requires a valid issuer signature on (V^*_U, c'). Since the honest issuer only signed $(V_U, \mathsf{com}_{\mathsf{att}})$, producing a valid σ^*_{cred} constitutes a forgery against the issuer's SPS scheme [Gro15].

Case 2: Credential Data Match ($(V^*_U, c') = (V_U, \mathsf{com}_{\mathsf{att}})$). Since $(V^*_U, c') = (V_U, \mathsf{com}_{\mathsf{att}})$, the vouch verification checks a signature σ^*_{st} relative to the honest user key $V_U = h^{v_U}$ on the message tuple

$$m' = (g^{\mathsf{st}'}, g^{z_{\mathsf{idx}'}(\tau)}), \quad \text{where } z_{\mathsf{idx}'}(X) = \prod_{i \in \mathsf{idx}'}(X - i).$$

Let

$$S_Q = \{(g^{\mathsf{st}}, g^{z_{\mathsf{idx}}(\tau)}) \mid (\mathsf{st}, \mathsf{idx}, \mathsf{com}_{\mathsf{att}}) \in Q,\ z_{\mathsf{idx}}(X) = \prod_{i \in \mathsf{idx}}(X - i)\}$$

be the set of messages signed by the honest user oracle. One of the following must hold:

Sub-case 2a: New Message ($m' \notin S_Q$). If $m' \notin S_Q$, then σ^*_{st} is a valid signature on a message that was never queried to the signing oracle. This constitutes a forgery against the user's SPS scheme [Gro15].

Sub-case 2b: Re-used Message ($m' \in S_Q$). Suppose $m' \in S_Q$. Then there exists $(\mathsf{st}, \mathsf{idx}, \mathsf{com}_{\mathsf{att}}) \in Q$ such that

$$(g^{\mathsf{st}'}, g^{z_{\mathsf{idx}'}(\tau)}) = (g^{\mathsf{st}}, g^{z_{\mathsf{idx}}(\tau)}).$$

Because the attack is successful, $(\mathsf{st}', \mathsf{idx}') \neq (\mathsf{st}, \mathsf{idx})$ (since $(\mathsf{st}', \mathsf{idx}', c') \notin Q$ and $c' = c = \mathsf{com}_{\mathsf{att}}$). From $g^{\mathsf{st}'} = g^{\mathsf{st}}$ we get $\mathsf{st}' = \mathsf{st}$, hence $\mathsf{idx}' \neq \mathsf{idx}$. Therefore $z_{\mathsf{idx}'}(X) \neq z_{\mathsf{idx}}(X)$ but $z_{\mathsf{idx}'}(\tau) = z_{\mathsf{idx}}(\tau)$, which breaks the binding property of the KZG polynomial commitment scheme.

In all cases, a successful adversary implies breaking either the unforgeability of the SPS scheme (Case 1, Sub-case 2a) or the binding of the polynomial commitment (Sub-case 2b). □

Lemma 3 (Vouch Extractability). *Construction 1 satisfies vouch extractability against generic adversaries.*

Proof. Let $\mathcal{A}$ be any PPT generic adversary that outputs $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$, where $\pi = (\sigma_{\mathsf{st}}, V_U, \sigma_{\mathsf{cred}}, \pi^1, \pi^2)$. We construct an extractor Ext as follows:

1. Run $\mathcal{A}$ and obtain $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$.
2. If $\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) = 0$, output $\perp$.
3. Otherwise, run the extractability algorithm for the hiding KZG PCS (Theorem 4) on the algebraic transcript defined by (c, π^1, π^2). Except with negligible probability, this returns $(\hat{f}(X), \hat{r})$ such that
$$c = h^{\hat{f}(\tau)} \cdot (h^{\eta})^{\hat{r}}.$$
4. Define $\mathsf{att}^*[i] := \hat{f}(i)$ for each $i \in [\ell]$, and output $(\mathsf{att}^*, \hat{r})$.

By definition of att^*, the unique degree-$< \ell$ polynomial consistent with the vector att^* is exactly $\hat{f}(X)$. By definition of the commitment algorithm in Construction 1,
$$\mathsf{com}(\mathsf{crs}, \mathsf{att}^*; \hat{r}) = h^{f_{\mathsf{att}^*}(\tau)} \cdot (h^{\eta})^{\hat{r}} = h^{\hat{f}(\tau)} \cdot (h^{\eta})^{\hat{r}} = c.$$
Hence, whenever $\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) = 1$, the extractor outputs (att^*, r) with $r = \hat{r}$ and $\mathsf{com}(\mathsf{crs}, \mathsf{att}^*; r) = c$, except with negligible probability. Therefore,
$$\Pr[\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) = 1 \ \wedge \ \mathsf{com}(\mathsf{crs}, \mathsf{att}^*; r) \neq c] \leq \mathsf{negl}(\lambda).$$
□

Lemma 4 (Vouch Simulatability). *Construction 1 satisfies vouch simulatability against generic adversaries.*

Proof. Let $\mathsf{td} = (\tau, \eta)$ be the trapdoor output by Setup. In the simulation argument, we give td to Sim (suppressed in the oracle notation $\mathsf{O}_{\mathsf{Sim}}(\mathsf{isk}, \cdot, \cdot)$) along with the revealed attributes att′, their indicesidx, and the statement st. We define a stateful simulator Sim that initializes one synthetic credential and answers all vouch queries using trapdoor-simulated KZG openings.

Initialization: Sample $\tilde{v}_U, \tilde{r}_{\mathsf{att}} \leftarrow\!\!\$\ \mathbb{F}$ and a polynomial $\tilde{f}(X) \in \mathbb{F}[X]$ of degree $< \ell$. Set $\tilde{V}_U := h^{\tilde{v}_U}$ and
$$\tilde{\gamma} := \tilde{f}(\tau) + \eta \tilde{r}_{\mathsf{att}}, \qquad \tilde{c} := h^{\tilde{\gamma}}.$$

Compute
$$\tilde{\sigma}_{\mathsf{cred}} \leftarrow \mathsf{Sign}(\mathsf{isk}, (\tilde{V}_U, \tilde{c})).$$

Store state $(\tilde{v}_U, \tilde{f}, \tilde{\gamma}, \tilde{V}_U, \tilde{c}, \tilde{\sigma}_{\mathsf{cred}})$.

Query $(\mathsf{idx}, \mathsf{att}', \mathsf{st})$**:** Let
$$z_{\mathsf{idx}}(X) = \prod_{i \in \mathsf{idx}} (X - i),$$
and let $\tilde{u}_{\mathsf{idx}}(X)$ be the unique degree-$< |\mathsf{idx}|$ polynomial such that $\tilde{u}_{\mathsf{idx}}(i) = \tilde{\mathsf{att}}[i]$ for all $i \in \mathsf{idx}$. Note that the relevant positions of att can be obtained from att′. Define
$$\zeta_{\mathsf{idx}} := z_{\mathsf{idx}}(\tau), \qquad \mu_{\mathsf{idx}} := \tilde{u}_{\mathsf{idx}}(\tau).$$

Sample $a_{\mathsf{idx}} \leftarrow\!\!\$\ \mathbb{F}$ and set
$$b_{\mathsf{idx}} := (\tilde{\gamma} - \mu_{\mathsf{idx}} - a_{\mathsf{idx}} \zeta_{\mathsf{idx}})/\eta, \qquad \tilde{\pi}^1 := g^{a_{\mathsf{idx}}}, \qquad \tilde{\pi}^2 := g^{b_{\mathsf{idx}}},$$

and
$$\tilde{\sigma}_{\mathsf{st}} \leftarrow \mathsf{Sign}(\tilde{v}_U, (g^{\mathsf{st}}, g^{\zeta_{\mathsf{idx}}})).$$

Return
$$(\tilde{c}, \tilde{\pi}), \quad \tilde{\pi} = (\tilde{\sigma}_{\mathsf{st}}, \tilde{V}_U, \tilde{\sigma}_{\mathsf{cred}}, \tilde{\pi}^1, \tilde{\pi}^2).$$

By construction of b_{idx}, each simulated opening satisfies

$$e(g, \tilde{c}/h^{\mu_{\mathsf{idx}}}) = e(\tilde{\pi}^1, h^{z_{\mathsf{idx}}(\tau)}) \cdot e(\tilde{\pi}^2, h^{\eta}),$$

so every response is accepting. For fixed synthetic state $(\tilde{f}, \tilde{r}_{\mathsf{att}})$, the above opening distribution is identical to honest batch opening: in the honest algorithm,

$$a_{\mathsf{idx}} = \eta s + \tilde{q}_{\mathsf{idx}}(\tau), \qquad b_{\mathsf{idx}} = \tilde{r}_{\mathsf{att}} - \mu_{\mathsf{idx}} - s\zeta_{\mathsf{idx}},$$

with $\tilde{q}_{\mathsf{idx}}(X) = (\tilde{f}(X) - \tilde{u}_{\mathsf{idx}}(X))/z_{\mathsf{idx}}(X)$ and $s \leftarrow\$ \mathbb{F}$. Since a_{idx} is uniform in $\mathbb{F}$ and

$$\tilde{\gamma} - \mu_{\mathsf{idx}} = \zeta_{\mathsf{idx}}\tilde{q}_{\mathsf{idx}}(\tau) + \eta\tilde{r}_{\mathsf{att}},$$

this is exactly

$$b_{\mathsf{idx}} = (\tilde{\gamma} - \mu_{\mathsf{idx}} - a_{\mathsf{idx}}\zeta_{\mathsf{idx}})/\eta.$$

Hence trapdoor simulation is perfect for that credential. Now we show that adversary's view when interating with the real oracle for fixed att is statistically indistinguihsable from the simulated one via a sequence of hybrids. First replace honest openings by the trapdoor-simulated openings for the same credential; this is a perfect change by the argument above. Then replace the committed polynomial f_{att} by independent $\tilde{f}$. The resulting change in the joint distribution of (c, π^1, π^2) over adaptive queries is negligible by the statistical hiding guarantee of hiding KZG (Theorem 4). The two SPS-signature components remain identically distributed given their signed messages: σ_{cred} is always a valid randomized SPS signature on (V_U, c), and σ_{st} is on $(g^{\mathsf{st}}, g^{z_{\mathsf{idx}}(\tau)})$, where z_{idx} depends only on idx. Therefore

$$\left|\Pr\left[\mathcal{A}^{\mathsf{O}_{\mathsf{Vouch}}(\cdot,\mathsf{att},\cdot,\mathsf{cred})}(\mathsf{ipk}) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{O}_{\mathsf{Sim}}(\mathsf{isk},\sigma,\cdot,\cdot)}(\mathsf{ipk}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

which is exactly vouch simulatability. □

9 Blackbox Proofs of Personhood

Groth-Sahai Proofs. Groth-Sahai [GS08] gave a NIWI proof for pairing product relations of the form

$$\prod_{i=1}^{n} e(A_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{c_{ij}} = t_T,$$

for constants $A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, c_{ij} \in \mathbb{Z}_p$, and $t_T \in \mathbb{G}_T$. When $t_T = 1_T$, this yields a NIZK proof because $\mathcal{X} = 1$ and $\mathcal{Y} = 1$ are valid solutions and therefore the simulator can use this witness in the simulation. Alternatively, if we can express $t_T = \prod_i e(p_i, q_i)$ for some known p_i's and q_i's, then by introducing additional witness elements and constraints of the form

$$\mathcal{Z}_i^{\delta} \cdot q_i^{-\delta} = 1_2$$

and rewriting the pairing product equation as

$$\prod_{i=1}^{n} e(A_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{c_{ij}} \prod_{i} e(p_i, \mathcal{Z}_i) = 1_T,$$

we can again obtain a NIZK using a similar strategy by using the canonical *zero* witness.

We now describe the crs and commitment scheme used in an instantiation of the Groth-Sahai proof system under the SXDH assumption. Let g and h be randomly sampled generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We omit how the proof is generated and verified as it is not relevant for the task at hand – building a commit and prove proof system that is compatible with a general purpose SNARK and a Groth-Sahai proof system. Looking ahead, this will allow us to treat a KZG commitment to the vector of field elements as a witness element in the Groth-Sahai proof system – thus allowing us to prove knowledge of a signature on it AND simultaneously treat the vector of field elements as witness elements in a general purpose zkSNARK.

- $\mathsf{Setup}(1^\lambda)$: Sample $\alpha_1, \alpha_2, t_1, t_2 \leftarrow\$ \mathbb{F}$ and compute $u_1 = (g, g^{\alpha_1})$, and $v_1 = (h, h^{\alpha_2})$. If
  - **Perfectly Binding:** $u_2 = (g^{t_1}, g^{\alpha_1 t_1}) = u_1^{t_1}$ and $v_2 = (h^{t_2}, h^{\alpha_2 t_2}) = u_2^{t_2}$
  - **Perfectly Hiding:** $u_2 = (g^{t_1}, g^{\alpha_1 t_1 - 1}) = u_1^{t_1} \cdot (1, g)^{-1}$ and $v_2 = (h^{t_2}, h^{\alpha_2 t_2 - 1}) = u_2^{t_2} \cdot (1, h)^{-1}$

  Output $\mathsf{crs} = (u_1, u_2, v_1, v_2)$.

- $\mathsf{Commit}(\mathsf{crs}, \cdot)$: To commit to an element
  - $\mathcal{X} \in \mathbb{G}_1$: Sample $r_1, r_2 \leftarrow\$ \mathbb{F}$ and output $\mathsf{com} = u_1^{r_1} \cdot u_2^{r_2} \cdot (1, \mathcal{X})$
  - $\mathcal{Y} \in \mathbb{G}_2$: Sample $r_1, r_2 \leftarrow\$ \mathbb{F}$ and output $\mathsf{com} = v_1^{r_1} \cdot v_2^{r_2} \cdot (1, \mathcal{Y})$

Remark 8. *Observe that when using a perfectly binding commitment key, commitments to $\mathcal{X} \in \mathbb{G}_1$ are of the form* $\mathsf{com} = (g^{r_1 + r_2 t_1}, g^{\alpha_1 (r_1 + r_2 t_1)} \cdot \mathcal{X})$*, which can be viewed as an ElGamal encryption of $\mathcal{X}$. On the other hand, when using a perfectly hiding commitment key, because u_1 and u_2 are linearly independent, there always exists randomness (r_1', r_2') to open a given commitment* com *to any message $\mathcal{X}' \in \mathbb{G}_1$.*

In both cases, the verification of the commitment is pedersen-like *in that it is essentially a multi-exponentiation. [CFQ19] showed that a large class of zkSNARKS support commit-and-prove techniques for pedersen-like commitments. Thus allowing us to treat a KZG commitment as a witness element in a pairing product relations, AND simultaneously treat the evaluations of the underlying polynomial (on a chosen domain) as witness elements in a zkSNARK.*

9.1 Instantiating Proofs of Personhood

We now show how the Vouchable Credential scheme from Construction 1 instantiates the generic Proof of Personhood construction from Fig. 3.

Trusted Setup. The Setup algorithm for the Vouchable Credential scheme establishes the KZG CRS and SPS public parameters $(g, h, Y_1, Y_2, W_1, W_2)$. This is run once as a trusted setup before any issuer initialization.

Issuer Initialization. The Init algorithm of Construction 1 directly instantiates the issuer key generation in Setup of Fig. 3. Each issuer $I^{(i)}$ samples $v_I \leftarrow\$ \mathbb{F}$ and publishes $V_I = g^{v_I} \in \mathbb{G}_1$ as their public key.

PHC Issuance. The IssuePHC protocol maps directly to the Issue protocol of Construction 1. The PRF-derived key $k_U = \mathsf{PRF.Eval}(\mathsf{K}, \mathsf{ipk})$ is simply treated as an additional attribute in the attribute vector $\mathsf{att} = k_U \| \mathsf{att}_U$. The user proves knowledge of their PRF key using the NIZK $\pi_{\mathsf{NIZK,PRF}}$, which can be instantiated as a standard Schnorr proof if the PRF is instatiated as $H(x)^{\mathsf{sk}}$, where $H : \{0,1\}^* \to \mathbb{G}_1$. The output of the PRF can then be hashed into a field element (and therefore become an attribute). The issuer then signs $(V_U, \mathsf{com}_{\mathsf{att}})$ using the [Gro15] scheme.

VRC Issuance and Verification. The IssueVRC and VerifyVRC protocols instantiate directly via the Vouch and Verify algorithms of Construction 1. The VRC issuer calls Vouch to produce a signature on the statement and revealed attributes, and the VRC verifier calls Verify to check the signature.

ShowVRC and VerifyShow. The language $\mathcal{L}_{\mathsf{ShowVRC}}$ requires proving:

1. Valid VRCs for each statement (pairing product equations)
2. PRF key ownership: $k_j' = \mathsf{PRF.Eval}(\mathsf{K}, k_j)$ for all j
3. Predicate satisfaction: $f(\mathsf{att}_1, \mathsf{st}_1, \ldots, \mathsf{att}_N, \mathsf{st}_N) = 1$

VRC Verification (Groth-Sahai). Observe that the Verify algorithm of Construction 1 is purely algebraic—it consists only of pairing product equations. Specifically, verification checks the following:

- **Credential signature verification:**

$$e(R, S) = e(g, Y_1) \cdot e(V_I, h)$$
$$e(R, T_1) = e(g, V_U) \cdot e(V_I, Y_1)$$
$$e(R, T_2) = e(g, \mathsf{com}_{\mathsf{att}}) \cdot e(V_I, Y_2)$$

- **KZG divisibility proof:**

$$e(g^{\prod_{i \in \mathsf{idx}}(\tau - \mathsf{att}'[i])}, \pi_{\mathsf{idx}}) = e(g, \mathsf{com}_{\mathsf{att}})$$

- **Vouch signature verification:**

$$e(S_{\mathsf{st}}, R_{\mathsf{st}}) = e(W_1, h) \cdot e(g, V_U)$$
$$e(T_{\mathsf{st},1}, R_{\mathsf{st}}) = e(g^{\mathsf{st}}, h) \cdot e(W_1, V_U)$$
$$e(T_{\mathsf{st},2}, R_{\mathsf{st}}) = e(g^{\prod_{i \in \mathsf{idx}}(\tau - \mathsf{att}'[i])}, h) \cdot e(W_2, V_U)$$

These pairing product equations fit precisely into the Groth-Sahai proof framework. By committing to the witness elements $(V_U, \mathsf{com}_{\mathsf{att}}, \sigma_{\mathsf{cred}}, \sigma_{\mathsf{st}}, \pi_{\mathsf{idx}}, g^{\prod_{i \in \mathsf{idx}}(\tau - \mathsf{att}'[i])})$ using Groth-Sahai commitments, we obtain a zero-knowledge proof of knowledge of a valid VRC.

To handle the remaining constraints (over field elements) – PRF evaluation, nullifier revocation checks, and the predicate f – we use a commit-and-prove SNARK as described in [CFQ19]. The key observation is that the Groth-Sahai commitment scheme which commit to KZG commitments, effectively form a pedersen-like commitment and are thus compatible with the commit-and-prove framework. This allows us to treat the coefficient of the polynomial underlying the KZG commitment $g^{\prod_{i \in \mathsf{idx}}(\tau - \mathsf{att}'[i])}$ as witness elements in the commit-and-prove SNARK.

10 Evaluation

We implement the constructions from section 8 and section 9 to demonstrate their practicality. However, one wonders whether a bespoke construction is required at all for these schemes, and we study this question by implementing the algorithms using general purpose zkSNARKs, specifically Groth16 [Gro16], with non-black-box cryptography instantiations. Although there a large number of off-the-shelf SNARK systems, we only experiment with Groth16 given that we find orders of magnitude better efficiency with our blackbox construction. Our code can be found at `https://github.com/LF-Decentralized-Trust-labs/zkbk`.

General-purpose zkSNARKs. While one could just as well use standard signature algorithms such as RSA or ECDSA for credential issuance and vouching, and prove knowledge of valid signatures by inlining the signature verification algorithms within the SNARK circuit, our goal is to study the best possible performance of the general-purpose SNARK approach. To that end, we avoid using non-native arithmetic within the SNARK circuit, by using the following choices for concrete cryptographic instantiations. We use the bls-12-381 [Bow17] pairing curve for Groth16. Signatures for credential issuance and vouching are implemented using Schnorr, instantiated using the JubJub elliptic curve [HBHW26] that is built over the bls-12-381 scalar field. We make use of Merkle trees, for committing to a list of attributes, where the Merkle root is signed under Schnorr – the hashing method in the Merkle tree is instantiated with Pedersen hashing [HBHW22], employing a 4-bit window Pedersen hash that maps bit sequences to compressed elliptic curve points, allowing PRF outputs to be field elements in the SNARK's scalar field.

Our Blackbox Construction For our blackbox constructions in sec.tion 8 and section 9, we use the bls-12-381 pairing curve to implement the structure-preserving signature [Gro15] and Groth-Sahai proofs [GS08].

Table 1: Mean Running Times for Vouchable Credentials and Personhood credentials.

Operation	Blackbox Construction	General-Purpose SNARK
Issuer key generation	0.92 ms	0.12 ms
User key generation	0.50 ms	0.12 ms
Credential issuance	2.87 ms	0.06 ms
Vouch generation	0.713 ms	0.06 ms
Personhood proof generation	33.95 ms	2.30 s
Personhood proof verification	100.54 ms	3.5 ms

Table 2: Comparison of our BlackBox construction vs. General-purpose SNARK over an increasing number of vouches

	CRS Size (MB)		Prover Time (s)		Verifier Time (s)	
Vouches	BlackBox	SNARK	BlackBox	SNARK	BlackBox	SNARK
1	0.0134	119	0.03	2.28	0.10	0.003
5	0.0134	668	0.16	11.81	0.50	0.003
10	0.0134	1300	0.33	23.80	1.00	0.003
25	0.0134	3080	0.84	61.96	2.51	0.003
50	0.0134	6150	1.69	162.14	5.03	0.003
100	0.0134	12,300	3.40	427.14	10.05	0.003

Experimental results. We implement all algorithms, under both general-purpose and blackbox approaches, using the elliptic curve and SNARK libraries within the arkworks [ac22] framework. All experiments were conducted on a MacBook Pro with a 12-core Apple M4 Pro and 32 GB of memory (though we only use 1 CPU core for all experiments). Benchmarks use Criterion [Hei24] with sufficient warm-up iterations and sample sizes to ensure statistical reliability – for each metric, we launch 20 runs and report the average numbers.

Microbenchmarks. Table 1 contains the results on our microbenchmarks, where we measure the running time of each algorithm on a single input. That is, we have a single issuer, who generates a credential for a single user; that user goes on to vouch for another user; finally, a personhood proof is generated over a single vouch, and then verified. Not surprisingly, the use of Schnorr signatures allows for much faster key generation and signing compared to structure-preserving signatures in [Gro15]. However, we argue that it is an appropriate tradeoff, as those operations are relatively infrequent, and moreover, the difference in running time is barely noticeable to a human user. The key distinction is in the tradeoff between proof generation and proof verification. Expectedly, Groth-Sahai proofs result in significantly faster proof generation, but at the cost of slower verification. However, as we see below, once we prove personhood using multiple vouches, as real applications require, we find a scaling challenge, described next.

Scaling. As proofs are constructed over increasing number of vouches[2], we naturally find that the CRS size, and proving and verification time may also grow with that parameter. We present the observations in Table 2. While there are no surprises here, it is worth discussing the scaling bottlenecks. Beyond 10 vouches, the Groth16 proving key is upwards of 1 GB, which presents a barrier for mobile proving – naturally, it also leads to gigabytes of RAM usage, which can be problematic on resource-constrained mobile devices. On the other hand, the Groth-Sahai proof system in the blackbox construction requires a fixed size commitment key of size 1344 bytes. Similarly, the proving time for general SNARK approach is upwards of 10 seconds for even 5 vouches; the blackbox construction is two orders of magnitude lower, which is paramount for

[2]In most applications, security is often strengthened by increasing the number of vouches during a *show*.

user experience. The tradeoff is that we find a slower verification. While this can be a deterrence in some applications, we find it acceptable in our setting for these reasons: 1) the prover time for Groth16 grows much faster than the verifier time for the blackbox scheme, making the blackbox scheme far more practical of the two; 2) verifiers for personhood proofs are typically service providers, which can afford server-grade hardware[3].

Acknowledgement.

Generative AI tools were used to assist in editing and refining portions of the text. AI-assisted coding tools were also used during development of the implementation evaluated in this work. All research contributions and technical claims remain the responsibility of the authors.

[3]Our experiments were only allowed a single CPU core, but the verification in our blackbox construction is easily parallelizable to make use of multiple CPU cores. This can be valuable when there is an assymetry in the computation resources of users and verifiers.

References

[ABF+25] Amine Allouah, Omar Besbes, Josué D Figueroa, Yash Kanoria, and Akshit Kumar. What is your ai agent buying? evaluation, biases, model dependence, & emerging implications for agentic e-commerce. *arXiv preprint arXiv:2508.02630*, 2025. 63

[ac22] arkworks contributors. `arkworks` zksnark ecosystem, 2022. 55

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010. 14

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Cham, August 2018. 13

[Bel26] Ashley Belanger. Discord faces backlash over age checks after data breach exposed 70,000 ids, February 2026. 4

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001. 15

[BFKT24] Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. The brave new world of global generic groups and UC-secure zero-overhead SNARKs. In *TCC 2024, Part I*, LNCS, pages 90–124. Springer, Cham, November 2024. 15

[BID20] Emanuele Bellini, Youssef Iraqi, and Ernesto Damiani. Blockchain-based distributed trust and reputation management systems: A survey. *IEEE Access*, 8:21127–21151, 2020. 62

[Bil26] Kembo Bill. The worldcoin autopsy: A case study in the failure of sovereign ai containment, February 2026. 4

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001. 16

[Bow17] Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction. `https://electriccoin.co/blog/new-snark-curve/`, March 2017. 54

[Bur25] Chris Burt. 'mdl don't phone home': digital id experts sound alarm over privacy capability, June 2025. 4

[CCoHS25] U.S. Customs, Border Protection (CBP), and Department of Homeland Security. Agency information collection activities; revision; arrival and departure record (form i-94) and electronic system for travel authorization (esta), December 2025. 64

[Cen23] Electronic Privacy Information Center. Epic statement on privacy risks of worldcoin, 2023. 4

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019. 13, 15, 53, 54

[CGM16] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 499–530. Springer, Berlin, Heidelberg, August 2016. 13

[CWH$^+$25] Alan Chan, Kevin Wei, Sihao Huang, Nitarshan Rajkumar, Elija Perrier, Seth Lazar, Gillian K Hadfield, and Markus Anderljung. Infrastructure for ai agents. *arXiv preprint arXiv:2501.10114*, 2025. 63

[DGH$^+$25] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7):1–36, 2025. 63

[DOL$^+$25] Gelei Deng, Haoran Ou, Yi Liu, Jie Zhang, Tianwei Zhang, and Yang Liu. Oedipus: LLM-enchanced reasoning CAPTCHA solver. In *ACM CCS 2025*, pages 6–20. ACM Press, November 2025. 4

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. 20

[Fou20a] The Linux Foundation. Kernel developer pgp keyring, 2020. 63

[Fou20b] The Linux Foundation. The kernel.org keysigning map, 2020. 63

[Fou20c] The Linux Foundation. Pdf of the kernel.org keysigning map, 2020. 63

[Fou26] World Foundation. Worldcoin whitepaper, 2026. 4, 8

[Fs24] Matteo Frigo and abhi shelat. Anonymous credentials from ECDSA. Cryptology ePrint Archive, Report 2024/2010, 2024. 62

[ftPS25] Bavaria Data Protection Authority for the Private Sector. Orders regarding the iris-codes, February 2025. 4

[GG21] Stan Gurtler and Ian Goldberg. Sok: Privacy-preserving reputation systems. *Proceedings on Privacy Enhancing Technologies*, 2021. 62

[GLM$^+$24] Scott Griffy, Anna Lysyanskaya, Omid Mir, Octavio Perez-Kempner, and Daniel Slamanig. Delegatable anonymous credentials from mercurial signatures with stronger privacy. In *ASIACRYPT 2024, Part II*, LNCS, pages 296–325. Springer, Singapore, December 2024. 62

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 5

[Goo25] Dan Goodin. Discord says hackers stole government ids of 70,000 users, October 2025. 4

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Berlin, Heidelberg, December 2007. 18

[Gro15] Jens Groth. Efficient fully structure-preserving signatures for large messages. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 239–259. Springer, Berlin, Heidelberg, November / December 2015. 14, 46, 47, 48, 49, 50, 53, 54, 55

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016. 54

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008. 14, 18, 49, 52, 54, 62

[Has] Mitchell Hashimoto. Vouch: A community trust management system. 5, 62

[HBHW22] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification, 2022. 54

[HBHW26] Daira-Emma Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash Protocol Specification. `https://zips.z.cash/protocol/protocol.pdf`, 2026. Section 5.4.9.3: Jubjub. 54

[Hei24] Brook Heisler. Criterion.rs: Statistics-driven benchmarking library for rust, 2024. 55

[Jam25] Sam James. Faq on the xz-utils backdoor (cve-2024-3094), 2025. 4

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004. 15

[Kap25] Matt Kapko. The north korea worker problem is bigger than you think, March 2025. 5

[Kas24] Kaspersky. Xz backdoor story – initial analysis, 2024. 4

[KS25] Darya Kaviani and Srinath Setty. Vega: Low-latency zero-knowledge proofs over existing credentials. Cryptology ePrint Archive, Paper 2025/2094, 2025. 62

[KSGM03] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003. 62

[KT24] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments: T. kohrita, p. towa. *Journal of Cryptology*, 37(4):38, 2024. 19, 20

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. 14, 19

[Lak25] Ravie Lakshmanan. Akirabot targets 420,000 sites with openai-generated spam, bypassing captcha protections, April 2025. 4

[Mar25] Bernard Marr. The 8 ai agent trends for 2026 everyone must be ready for now, October 2025. 62

[MBG$^+$23] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 30–44. ACM Press, November 2023. 62

[MN25] Maggie Miller and Dana Nickel. Tech companies have a big remote worker problem: North korean operatives, May 2025. 5

[MSM23] Omid Mir, Daniel Slamanig, and René Mayrhofer. Threshold delegatable anonymous credentials with controlled and fine-grained delegation. *IEEE Transactions on Dependable and Secure Computing*, 21(4):2312–2326, 2023. 62

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994. 15

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 17

[oSS25] Chinese Ministry of State Security. These things may be lost, but the ”password” cannot be changed., August 2025. 4

[Out26] Goldman Sachs 2026 Outlook. What to expect from ai in 2026: Personal agents, mega alliances, and the gigawatt ceiling, January 2026. 62

[pe25] privacy ethereum. zkid. `https://github.com/privacy-ethereum/zkID`, 2025. 62

[PPZ24] Christian Paquin, Guru-Vamsi Policharla, and Greg Zaverucha. Crescent: Stronger privacy for existing credentials. Cryptology ePrint Archive, Report 2024/2013, 2024. 62

[PVW24] Andreas Plesner, Tobias Vontobel, and Roger Wattenhofer. Breaking recaptchav2. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1047–1056. IEEE, 2024. 4

[RCER25] Marco De Rossi, Davide Crapis, Jordan Ellis, and Erik Reppel. Erc-8004: Trustless agents, August 2025. 62

[Reu24a] Reuters. Hong kong regulator directs worldcoin to cease operations citing privacy concerns, May 2024. 4

[Reu24b] Reuters. Worldcoin must delete all iris scan data, watchdog says, December 2024. 4

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. 15

[Sin25] Evie Kim Sing. Overlooked “phone home” surveillance capabilities in mdl standard trigger experts, June 2025. 4

[SJVS25] Rao Surapaneni, Miku Jha, Michael Vakoc, and Todd Segal. Announcing the agent2agent protocol (a2a), April 2025. 63

[SMH$^+$25] Tobin South, Samuele Marro, Thomas Hardjono, Robert Mahari, Cedric Deslandes Whitney, Dazza Greenwood, Alan Chan, and Alex Pentland. Authenticated delegation and authorized ai agents. *arXiv preprint arXiv:2501.09674*, 2025. 63

[SSG$^+$25] Georgios Syros, Anshuman Suri, Jacob Ginesin, Cristina Nita-Rotaru, and Alina Oprea. Saga: A security architecture for governing ai agentic systems. *arXiv preprint arXiv:2504.21034*, 2025. 63

[Ste93] Peter Steiner. On the internet, nobody knows you're a dog, July 1993. 4

[TCBM20] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in neural information processing systems*, 33:1633–1645, 2020. 63

[TLL$^+$25] Xiwen Teoh, Yun Lin, Siqi Li, Ruofan Liu, Avi Sollomoni, Yaniv Harel, and Jin Song Dong. Are CAPTCHAs still bot-hard? Generalized visual CAPTCHA solving with agentic vision language model. In *USENIX Security 2025*, pages 3747–3766. USENIX Association, August 2025. 4

[Tur50] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. 4

[Und] Undersigned. No phone home statement. 4

[vABHL03] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 294–311. Springer, Berlin, Heidelberg, May 2003. 4

[WPGS25] Yuntao Wang, Yanghe Pan, Shaolong Guo, and Zhou Su. Security of internet of agents: Attacks and countermeasures. *IEEE Open Journal of the Computer Society*, 2025. 63

[XZS$^+$26] Bin Xie, Tianyu Zheng, Rui Song, Shang Gao, and Bin Xiao. Re2creds: Reusable anonymous credentials from malleable NIZK and legacy signatures. Cryptology ePrint Archive, Paper 2026/119, 2026. 62

[ZGK$^+$17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880. IEEE Computer Society Press, May 2017. 19

[ZJBS24] Tingwei Zhang, Rishi D. Jha, Eugene Bagdasaryan, and Vitaly Shmatikov. Adversarial illusions in multi-modal embeddings. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. 63

[ZW26] Yu Zhang and Zongbin Wang. Toward verifiable privacy in decentralized identity: A formal framework for minimal disclosure and unlinkability. Cryptology ePrint Archive, Paper 2026/127, 2026. 62

A Related Work

In this section we outline additional related work.

Legacy Compatible Anonymous Credentials. There has been a recent line of work [Fs24, PPZ24, KS25, pe25] on using zkSNARKs to build anonymous credentials directly from *legacy* credentials issued in the format of standardized signatures (such as ECDSA/RSA) on a JSON/MDOC file. At a high-level the approach is to prove knowledge of a valid credential by showing that "I have a signature on a JSON file", parsing out relevant parts of the JSON file to *extract* attributes and then carrying out selective disclosure/revealing arbitrary functions of the attributes.

These works target an important but *orthogonal* problem where issuers do not need to make any modifications to their infrastructure. This is invaluable when relying issuers such as the DMV or Passport Issuers for credentials. Given that the *showing* of these anonymous credentials is a zero-knowledge proof, they can also be extended to build Vouchable Credentials (Section 6). We can then use our generic construction (Fig. 2) to build a proof of personhood. The generic construction is non-black box in the anonymous credential verifier so it would be quite inefficient. However, for a *practical* construction, one can start from Crescent [PPZ24], which has an *algebraic* verification only involving pairing product equations. Crescent can then be extended to build vouchable credentials as done in Section 8 which can then be combined with Groth-Sahai proofs [GS08] to build a proof of personhood.

Standard Anonymous Credentials and Delegatable Anonymous Credentials While there are too many anonymous credential schemes to mention here, there are a number of related anonymous credential schemes that are more relevant than most to us, including focused on decentralized identifiers (DIDs) [ZW26] and schemes that offer similar powerful functionalities [XZS$^+$26, GLM$^+$24, MSM23, MBG$^+$23]. Although techniques from these schemes are closely related to what we build here, these schemes do not allow us to directly build proofs of personhood and are essentially an incomparable primitive.

Decentralized Reputation Systems. In our work, we show how to *build* a reputation system, but we don't explicitly focus on how users should judge or build reputations. There has been extensive work on this topic, going back all the way to famous works like Eigentrust [KSGM03]. More recently, there have been many works on blockchain-based reputation systems [BID20] and privacy-preserving reputation systems [GG21]; we cite surveys for these because the overall amount of literature is quite large.

Other Protocols. One extremely interesting reputation protocol is the Ethereum Improvement Proposal 8004 (ERC 8004) [RCER25]. This standard proposes a trust metric for online agents on the blockchain, which can establish trust through reputation and validation. However, this system does not satisfy the privacy guarantees we desire. Very recently, Mitchell Hashimoto, the founder of Hashicorp, created a protocol called "Vouch" [Has] which is designed to combat low-quality, AI-generated contributions to open-source projects by requiring explicit social interaction and approval (which they also refer to as vouching) from trusted maintainers before a contributor can participate.

B Applications

In this subsection, we explain some of the applications of proofs of personhood and personhood credentials in more detail.

B.1 AI Agent Reputation

Many businesses expect that online commerce will become agentic in the not too distant future [Mar25, Out26]. In this world, users buying goods will have personal agents that interact with agents of vendors selling goods. The users' purchasing agents will negotiate with the sellers' agents and try to find the best deal that maximally meets the user's needs. Having an agent that can shop for you seems like a wonderful thing, until the possibility of malicious agents exploiting a buying agent is taken into account.

Unfortunately, it is extremely difficult to secure AI agents against malicious behavior or other malicious agents. There are a huge number of recent attacks in this space [ZJBS24, DGH+25, WPGS25], including attacks that directly exploit shopping agents [ABF+25]. It seems difficult to come up with meaningful defenses to attacks, as researchers almost continually break recently proposed defenses [TCBM20]. In the future, it seems unlikely that we will be able to rely on agents not being exploitable by malicious behavior.

So how can we handle this? The seemingly simplest way–and what we propose–that offers security while still preserving privacy and decentralization is to use a decentralized reputation system for AI agents. People (with personhood credentials) delegate their credentials to an agent. This agent can have a reputation (with vouches) based on its owner's reputation as well as its behavior. Sellers' agents will have similar reputations, which can be built through vouches after good transactions, and if either a buyer's or seller's agent misbehaves, its credentials can be revoked. This system would enable trustworthy agentic commerce, as malicious agents would be quickly excluded from the system.

Proofs of personhood (delegated from users or companies to agents) and the other tools we build in this paper can be used to construct such a system. As we have mentioned before, proofs of personhood enable exactly this decentralized reputation system which is needed for this application.

There have been a number of proposed constructions that aim to mitigate some of these problems [CWH+25, SMH+25, SSG+25], including the A2A protocol from Google [SJVS25], but none of these constructions simultaneously offer user privacy and decentralization like our system does, both of which we believe are essential for large-scale agentic commerce.

B.2 Open Source Maintainer Provenance

In the introduction, we mentioned the XZUtils attack and how proofs of personhood could have been used to foil that attack or at least make it significantly harder or more expensive to execute. We next want to elaborate on this concept.

As a starting point, we note that some prominent open source software projects already essentially run some sort of proof of personhood for their maintainers. For instance, the Linux kernel maintains a kernel maintainer web of trust system based on a GPG key ring [Fou20a]. To be included, kernel verifiers must personally verify each other and sign each others' GPG keys, usually in person but sometimes over video conference [Fou20b]. Typically prospective inductees present government ID proving that they are who they claim to be. In-person signing is common enough that the kernel.org team publishes a map of the (rough) physical locations of maintainers [Fou20c] so that people looking to be included in the web of trust can find someone to physically verify their identity. Maintainers count as "verified" if they have a path on the graph induced by this web of trust originating at Linus Torvalds.

This current system is suboptimal in many respects. For starters, in-person verification of government documents could be replaced by digital credentials, saving maintainers a significant amount of travel. GPG signings are less useful than vouches with attributes, because they don't carry any context. The system is also fully public and open, meaning that maintainers are potentially exposed to social engineering attacks based on the web of trust.

We can reconstruct this system using proofs of personhood and other tools in this paper to capture all (and more!) of the functionality of the existing Linux kernel web of trust, as well as provide significantly more privacy for the maintainers. The issuers in this system would be traditional issuers that provided government IDs as well as the Linux Foundation (which provides an LFID), and GPG signings would be replaced with VRCs. Such a system would have considerably less friction for maintainers as well as offer better privacy properties while still maintaining the overall functionality and goals of the system.

B.3 Decentralized Business Networks

There are many centralized social networks today, and these are sometimes very controversially used by law enforcement as variants of proofs of personhood [CCoHS25]. In a more benign use, users might want to be able to prove to potential employers that their credentials are legitimate and that they have certain attributes that are desirable for employment. There are social networks today like LinkedIn that aim to capture this functionality.

In our system, issuers could include employers (who could verify employment status of employees through credentials), universities (who could issue digital degrees), and governments (who could issue credentials for employment status). Users could obtain these credentials and then issue VRCs for both basic proofs of personhood and more complicated vouches (e.g. “person XYZ is a qualified cryptographer”). This sort of system could include a mobile application that simplified this process of connection, and all of this data could be used for job applications. The first step of the application process could just be a proof that the qualifications for the job are met.

Since such a system would be decentralized and privacy-preserving, it would also have the benefit of being significantly more resistant to government abuse than centralized systems.

C Generic Construction of Vouchable Credentials

In this section we provide a generic transformation that upgrades the security of a regular credential scheme to a vouchable credential scheme. The only requirement from this underlying scheme is that there is a verification algorithm that verifies the issued credentials against *all* attributes with respect to the issuer public key. The transformation uses tag-based SE-NIZK (Definition 5) and non-interactive commitment schemes (Definition 3).

At a high level, the credential issuance for the vouchable credential will be identical to that of the underlying credential scheme. To produce a vouch for a statement st using a subset of revealed attributes att' for index set idx, one first generates a commitment c to all the attributes att, and then generates a NIZK proof that (i) c is a commitment to att; and (ii) $\mathsf{att}|_{\mathsf{idx}} = \mathsf{att}'$. The formal description of the transformation follows.

Credential Scheme. We start by defining a credential scheme, with minimal requirements. In particular, a credential scheme Cred is specified by the tuple $(\mathsf{Init}, \mathsf{Issue}_{I,U}, \mathsf{VerCred})$:

$\mathsf{Init}(1^\lambda, 1^\ell) \rightarrow (\mathsf{isk}, \mathsf{ipk})$**:** A randomized initialization algorithm that takes as input the security parameter, and a parameter ℓ specifying the upper bound on the number of attributes. It outputs issuer public and private keys ipk and isk.

$\mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle \rightarrow \mathsf{cred}/\bot$**:** An interactive protocol between an issuer I and user U. The issuer has input its secret issuer key isk whereas the user has as input the issuer public key and a set of ℓ attributes att. At the end of the protocol, the user gets as output a credential cred or the issuance fails with $\bot$.

$\mathsf{VerCred}(\mathsf{ipk}, \mathsf{att}, \mathsf{cred}) \rightarrow 0/1$**:** A deterministic algorithm which takes as input the issuer public key ipk, attributes att, and a credential cred. The algorithm outputs 0 or 1 to indicate whether it rejects or accepts the credentials for the specified attributes.

Definition 10 (Credential Scheme). *A credential scheme* $\mathsf{Cred} = (\mathsf{Init}, \mathsf{Issue}_{I,U}, \mathsf{VerCred})$ *must satisfy the following properties:*

Correctness. *For every* $\lambda \in \mathbb{N}$, $\ell \in \mathbb{N}$, *attributes* att*:*

$$\Pr\left[\ \bot \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle\ :\ (\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda, 1^\ell)\ \right] \le \mathsf{negl}(\lambda)$$

and

$$\Pr\left[\ \mathsf{VerCred}(\mathsf{ipk}, \mathsf{att}, \mathsf{cred}) = 0\ :\ \begin{array}{r}(\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda, 1^\ell)\\ \mathsf{cred} \leftarrow \mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle\end{array}\ \right] \le \mathsf{negl}(\lambda)$$

Unforgeability. *For any polynomial time adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}\mathsf{VerCred}(\mathsf{ipk},\mathsf{att}^*,\mathsf{cred}^*) = 1 \wedge \\ \mathsf{att}^* \notin Q_{\mathsf{attr}}\end{array} : \begin{array}{r}(\mathsf{isk},\mathsf{ipk}) \leftarrow \mathsf{Init}(1^\lambda,1^\ell) \\ (\mathsf{att}^*,\mathsf{cred}^*) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)}(\mathsf{ipk})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk},\cdot)$ is an oracle that executes the honest issuer protocol $\mathsf{Issue}_{I,U}$ on input attributes chosen by the adversary. Q_{attr} is defined as the set of all attribute vectors for which the adversary successfully completed the protocol with the issuance oracle.

C.1 Construction

Components. We specify the components underlying our construction.

- A credential scheme (Definition 10) $\mathsf{Cred} = (\mathsf{Cred.Init}, \mathsf{Cred.Issue}_{I,U}, \mathsf{Cred.VerCred})$.
- A commitment scheme (Definition 3) $\mathsf{Com} = (\mathsf{Com.Setup}, \mathsf{Com.com})$.
- A tag-based SE-NIZK (Definition 5) $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the language $\mathcal{L}_{\mathsf{cred}}$ (defined below).

Proof of Knowledge Language. The language $\mathcal{L}_{\mathsf{cred}}$ is defined below. We note that the language is parameterized by the crs from the commitment scheme, and the credential scheme. For ease of notation, we do not include this in the notation.

$$\mathcal{L}_{\mathsf{cred}} = \Big\{(\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx}) \;\mid \exists \mathsf{att}, r, \mathsf{cred} \text{ s.t. } \mathsf{att}|_{\mathsf{idx}} = \mathsf{att}' \wedge c = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}; r) \wedge$$
$$\mathsf{VerCred}(\mathsf{ipk}, \mathsf{att}, \mathsf{cred}) = 1\Big\}$$

Generic Vouchable Credential

Parameters: Security parameter 1^λ, attribute-length parameter 1^ℓ.

```
Setup(1^λ, 1^ℓ):
    1. $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda)$
    2. $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
    3. Output $\mathsf{pp} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{NIZK}})$, $\mathsf{crs} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{crs}_{\mathsf{NIZK}})$, and td.
Init(1^λ, 1^ℓ):
    1. Output $\mathsf{Cred.Init}(1^\lambda, 1^\ell)$
$\mathsf{Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle$:
    1. Run the underlying issuance protocol $\mathsf{Cred.Issue}_{I,U}$.
    2. Output cred or $\perp$ according to the output of the underlying protocol.
Vouch(ipk, st, att, idx, cred):
    1. Set $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$.
    2. Sample $r$ and compute commitment $c = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}; r)$.
    3. Set statement $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$, witness $\mathsf{w} = (\mathsf{att}, r, \mathsf{cred})$ and tag $t = \mathsf{st}$.
    4. Compute $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, t, \mathsf{x}, \mathsf{w})$.
    5. Output $(c, \pi)$.
Verify(ipk, st, att', idx, π, c):
    1. Set $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ and $t = \mathsf{st}$.
    2. Output $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, t, \mathsf{x}, \pi)$.
```

Extractor for vCred**.** Given an accepting tuple $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$, define

$$\mathsf{Ext}_{\mathsf{vCred}}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) : \ (\mathsf{att}^*, r^*, \mathsf{cred}^*) \leftarrow \mathsf{Ext}_{\mathsf{NIZK}}(\mathsf{crs}_{\mathsf{NIZK}}, t, \mathsf{x}, \pi),$$

where $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ and $t = \mathsf{st}$, then output (att^*, r^*).

Theorem 8. *Assuming the credential scheme, non-interactive commitment scheme and a tag-based SE-NIZK, the above construction is a vouchable credential (Definition 9).*

Proof. We prove each property by reduction to the corresponding security of the underlying primitives.

Correctness. For honestly generated $(\mathsf{isk}, \mathsf{ipk})$ and credential $\mathsf{cred} \leftarrow \mathsf{Cred.Issue}_{I,U}\langle(\mathsf{isk}, \mathsf{att}), (\mathsf{ipk}, \mathsf{att})\rangle$, correctness of Cred gives $\mathsf{Cred.VerCred}(\mathsf{ipk}, \mathsf{att}, \mathsf{cred}) = 1$. By construction, with $c = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}; r)$ and $\mathsf{att}' = \mathsf{att}|_{\mathsf{idx}}$, we have

$$(\mathsf{x}, (\mathsf{att}, r, \mathsf{cred})) \in R_{\mathcal{L}_{\mathsf{cred}}}$$

for $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ and tag $t = \mathsf{st}$. Therefore, by NIZK completeness, $\mathsf{Verify}(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c) = 1$ except negligible failure inherited from the underlying algorithms.

Unforgeability. Let $\mathcal{A}$ be any PPT adversary in the vouch unforgeability game of Definition 9. Suppose $\mathcal{A}$ outputs an accepting forgery $(\mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$ with non-negligible probability. We then construct an adversary $\mathcal{B}_{\mathsf{cred}}$ that breaks the unforgeability property of the underlying credential.

$\underline{\mathcal{B}_{\mathsf{cred}}^{\mathsf{O}_{\mathsf{Issue}}(\mathsf{isk}, \cdot)}(\mathsf{ipk})}$

1. $\mathcal{B}_{\mathsf{cred}}$ forwards issuance queries to its credential-issuance oracle and emulates the vouch game for $\mathcal{A}$.
2. On output $(\mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$, run $\mathsf{Ext}_{\mathsf{NIZK}}$ on (t, x, π) with $t = \mathsf{st}$ and $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ to obtain $(\mathsf{att}^*, r^*, \mathsf{cred}^*)$.
3. Output $(\mathsf{att}^*, \mathsf{cred}^*)$.

Define event $\mathsf{E}_{\mathsf{ext}}$ as: π verifies but extraction fails or returns a witness not in $R_{\mathcal{L}_{\mathsf{cred}}}$. If $\Pr\left[\ \mathsf{E}_{\mathsf{ext}}\ \right]$ is non-negligible, we obtain a break of the NIZK proof-of-knowledge property.

Conditioned on $\neg\mathsf{E}_{\mathsf{ext}}$, extraction returns a valid witness and hence

$$\mathsf{Cred.VerCred}(\mathsf{ipk}, \mathsf{att}^*, \mathsf{cred}^*) = 1, \quad \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}^*; r^*) = c, \quad \mathsf{att}^*|_{\mathsf{idx}} = \mathsf{att}'.$$

Therefore, whenever the vouch forgery is fresh (i.e., $\mathsf{att}^* \notin Q_{\mathsf{attr}}$), $(\mathsf{att}^*, \mathsf{cred}^*)$ is a valid credential forgery against Cred.

Vouch non-malleability. Fix adversary $\mathcal{A}$ in the non-malleability game of Definition 9. Consider the following hybrids.

Hyb_0: Real game with oracle $\mathsf{O}_{\mathsf{Vouch}}$.

Hyb_1: Same game, but each oracle response proof is replaced by

$$\pi \leftarrow \mathsf{NIZK.SimProve}(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}, t, \mathsf{x}),$$

with unchanged (t, x, c).

The only difference between Hyb_0 and Hyb_1 is replacement of real proofs by simulated proofs on the same (t, x). Hence, by multi-theorem computational zero-knowledge, $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

In Hyb_1, if $\mathcal{A}$ still wins with non-negligible probability, we build an adversary against tag-based weak simulation-extractability.

$\underline{\mathcal{B}_{\mathsf{SE}}^{\mathsf{SimProve}(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}, \cdot, \cdot)}(\mathsf{crs}_{\mathsf{NIZK}})}$

1. Emulate $\mathcal{A}$ and answer each vouch query $(\mathsf{st}, \mathsf{idx})$ by computing the same statement/tag pair (x, t) and querying the simulation oracle $\mathsf{SimProve}(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}, \cdot, \cdot)$.
2. Record queried pairs (t, x) and triples $(\mathsf{st}, \mathsf{idx}, c)$ in Q.
3. If $\mathcal{A}$ outputs an accepting $(\pi', c', \mathsf{st}', \mathsf{idx}')$ with $(\mathsf{st}', \mathsf{idx}', c') \notin Q$, output (t', x', π') where

$$t' = \mathsf{st}', \qquad x' = (\mathsf{ipk}, c', \mathsf{att}|_{\mathsf{idx}'}, \mathsf{idx}').$$

By construction, freshness of $(\mathsf{st}', \mathsf{idx}', c')$ implies freshness of (t', x') relative to simulation-oracle queries. Therefore any non-negligible success in this reduction breaks weak simulation-extractability, and vouch non-malleability follows.

Vouch extractability. Given an accepting tuple $(\mathsf{ipk}, \mathsf{st}, \mathsf{att}', \mathsf{idx}, \pi, c)$, set $\mathsf{x} = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ and $t = \mathsf{st}$, and run $\mathsf{Ext}_{\mathsf{NIZK}}$ to obtain $(\mathsf{att}^*, r^*, \mathsf{cred}^*)$. By NIZK proof-of-knowledge,

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, t, \mathsf{x}, \pi) = 1 \implies (\mathsf{x}, (\mathsf{att}^*, r^*, \mathsf{cred}^*)) \in R_{\mathcal{L}_{\mathsf{cred}}}$$

except negligible probability. Therefore

$$\mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}^*; r^*) = c$$

except negligible probability, and similarly $\mathsf{att}^*|_{\mathsf{idx}} = \mathsf{att}'$. This implies vouch extractability.

Vouch simulatability. In this construction, take $\sigma = \mathsf{td}$ from $\mathsf{NIZK.Setup}$. Define simulator Sim for oracle $\mathsf{O}_{\mathsf{Sim}}(\mathsf{isk}, \mathsf{td}, \cdot, \cdot)$ on query $(\mathsf{st}, \mathsf{idx})$:

$\underline{\mathsf{Sim}(\mathsf{isk}, \mathsf{td}, \mathsf{st}, \mathsf{idx}, \mathsf{att}')}$

1. Sample $\tilde{r}$, set $\tilde{c} = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, 0^\ell; \tilde{r})$, and set $\tilde{\mathsf{x}} = (\mathsf{ipk}, \tilde{c}, \mathsf{att}', \mathsf{idx})$ and $\tilde{t} = \mathsf{st}$.
2. Output $(\tilde{c}, \tilde{\pi})$, $\quad \tilde{\pi} \leftarrow \mathsf{NIZK.SimProve}(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}, \tilde{t}, \tilde{\mathsf{x}})$.

To prove indistinguishability from $\mathsf{O}_{\mathsf{Vouch}}$, use hybrids:

Hyb_0: Real oracle $\mathsf{O}_{\mathsf{Vouch}}$ on fixed $(\mathsf{att}, \mathsf{cred})$.

Hyb_1: Replace each real proof by SimProve on the same (t, x, c), where $x = (\mathsf{ipk}, c, \mathsf{att}', \mathsf{idx})$ and $t = \mathsf{st}$. By multi-theorem ZK, $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Hyb_2: Replace each commitment $c = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}; r)$ by $\tilde{c} = \mathsf{Com.com}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{att}_0; \tilde{r})$ where $\mathsf{att}_0 = 0^\ell$. By computational hiding of Com, $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Hyb_2 is exactly the oracle induced by Sim. By triangle inequality over the hybrid sequence, vouch simulatability holds.

Combining all items proves Theorem 8. $\square$